

# **24x7 Scheduler™ 2.4**

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

# Table of Contents

|  |           |
|--|-----------|
| <b>TABLE OF CONTENTS .....</b>                                       | <b>2</b>  |
| <b>CHAPTER 1: GETTING STARTED .....</b>                              | <b>11</b> |
| OVERVIEW .....   | 11        |
| Preface.....   | 11        |
| Introduction .....   | 11        |
| TO START 24x7 SCHEDULER EACH TIME WINDOWS STARTS .....               | 13        |
| RUNNING 24x7 SCHEDULER AS A WINDOWS NT SERVICE .....                 | 14        |
| RUNNING 24x7 SCHEDULER AS A WINDOWS 95/98 SERVICE .....              | 14        |
| JOB EXPLORER .....   | 15        |
| JOB TYPES .....  | 16        |
| FILE TYPES .....   | 17        |
| DRAG AND DROP INTERFACE .....  | 18        |
| SUPPORT FOR WINDOWS EXPLORER DRAG AND DROP .....                     | 18        |
| SEMAPHORE FILES .....  | 19        |
| EXECUTION LOGS .....   | 19        |
| CHANGES TO THE SYSTEM TIME.....                                      | 20        |
| WORKING WITH JOB DATABASE.....                                       | 20        |
| SENDING AND RECEIVING E-MAIL MESSAGES .....                          | 21        |
| Overview .....   | 21        |
| About Lotus Notes interface.....                                     | 22        |
| MAINTENANCE FOR HOLIDAYS, DATABASE PROFILES, AND REMOTE AGENTS ..... | 22        |
| <b>CHAPTER 2: INSTALLATION AND UNINSTALLATION .....</b>              | <b>24</b> |
| SYSTEM REQUIREMENTS .....  | 26        |
| <b>CHAPTER 3: DATABASE INTERFACES .....</b>                          | <b>27</b> |
| DATABASE PROFILES .....  | 28        |
| ODBC INTERFACE .....   | 29        |
| <b>CHAPTER 4: SCHEDULING JOBS.....</b>                               | <b>30</b> |
| JOB WIZARD .....   | 30        |
| JOB PROPERTIES.....  | 31        |
| General Properties.....  | 31        |
| Job Execution Properties .....                                       | 31        |
| Job Schedule and Triggers .....                                      | 34        |
| Notification Events and Actions.....                                 | 37        |
| Job Notification Options .....                                       | 37        |
| Notification Events.....   | 37        |
| Notification Actions.....  | 37        |
| MACRO-PARAMETERS .....   | 38        |
| ADDING NEW JOB .....   | 40        |
| DELETING JOB.....  | 40        |
| DISABLING/ENABLING JOB .....   | 41        |
| MODIFYING JOB DEFINITION AND SCHEDULE .....                          | 41        |
| MOVING JOB TO ANOTHER FOLDER .....                                   | 41        |
| COPYING JOB TO/FROM ANOTHER DATABASE .....                           | 42        |
| PROTECTING/UNPROTECTING JOB.....                                     | 43        |
| TESTING JOB EXECUTION .....  | 43        |
| ERROR MESSAGES .....   | 44        |
| STOPPING JOB EXECUTION.....  | 45        |

|  |           |
|--|-----------|
| CREATING JOB SHORTCUT .....  | 45        |
| SENDING KEYSTROKES TO OTHER PROGRAMS.....                            | 45        |
| TROUBLESHOOTING JOB EXECUTION .....                                  | 47        |
| Job Execution Statistics .....                                       | 47        |
| Disabling Timer .....  | 48        |
| Trace Features.....  | 48        |
| <b>CHAPTER 5: JOB INTERDEPENDENCIES.....</b>                         | <b>50</b> |
| OVERVIEW .....   | 50        |
| GRAPHICAL DEPENDENCIES EDITOR.....                                   | 50        |
| Adding New Job to Dependencies View .....                            | 51        |
| Deleting Job from Dependencies View .....                            | 52        |
| Arranging Jobs on Dependencies View .....                            | 53        |
| Adding New Dependency.....   | 53        |
| Deleting Dependency.....   | 53        |
| Printing Dependencies .....  | 54        |
| Using Zoom Tool.....   | 54        |
| <b>CHAPTER 6: FAIL-OVER MODE .....</b>                               | <b>55</b> |
| ABOUT FAIL-OVER MODE.....  | 55        |
| CONNECTION PARAMETERS FOR FAIL-OVER MODE AND FOR REMOTE AGENTS ..... | 56        |
| TESTING CONNECTION TO MASTER SCHEDULER .....                         | 58        |
| STARTING MASTER AND STANDBY SCHEDULERS.....                          | 58        |
| RESTRICTING ACCESS TO MASTER SCHEDULER .....                         | 59        |
| <b>CHAPTER 7: REMOTE AGENTS.....</b>                                 | <b>60</b> |
| ABOUT REMOTE AGENTS.....   | 60        |
| STARTING REMOTE AGENT .....  | 61        |
| RESTRICTING ACCESS TO REMOTE AGENTS .....                            | 61        |
| REMOTE AGENT PROFILES .....  | 62        |
| SAMPLE REMOTE AGENT SETUP .....                                      | 64        |
| SYNCHRONOUS AND ASYNCHRONOUS CONNECTIONS.....                        | 65        |
| <b>CHAPTER 8: DATABASE JOBS .....</b>                                | <b>66</b> |
| PREPARING TO USE YOUR DATABASE .....                                 | 66        |
| INSTALLING THE ODBC DRIVER OR NATIVE DATABASE DRIVER.....            | 66        |
| DEFINING THE ODBC DATA SOURCE .....                                  | 66        |
| CREATING DATABASE PROFILES.....                                      | 67        |
| TROUBLESHOOTING THE DATABASE CONNECTION.....                         | 67        |
| <b>CHAPTER 9: JOB AUTOMATION SCRIPTS.....</b>                        | <b>68</b> |
| <b>CHAPTER 10: LOGGING JOB EXECUTION.....</b>                        | <b>69</b> |
| <b>CHAPTER 11: STATUS REPORT.....</b>                                | <b>70</b> |
| USING WEB BROWSER TO SEE STATUS REPORT .....                         | 71        |
| <b>CHAPTER 12: EXCEPTION DATES .....</b>                             | <b>72</b> |
| <b>CHAPTER 13: EDITOR .....</b>                                      | <b>73</b> |
| <b>CHAPTER 14: LOG VIEWER .....</b>                                  | <b>79</b> |
| <b>CHAPTER 15: JOB MONITORING .....</b>                              | <b>80</b> |
| <b>CHAPTER 16: 24X7 REMOTE CONTROL.....</b>                          | <b>81</b> |
| ABOUT 24x7 REMOTE CONTROL.....                                       | 81        |
| MANAGING REMOTE JOBS AND CONFIGURATIONS .....                        | 82        |
| STARTING 24x7 REMOTE CONTROL .....                                   | 83        |
| <b>CHAPTER 17: 24X7 SCHEDULER API - EXTERNAL INTERFACES .....</b>    | <b>84</b> |

|  |           |
|--|-----------|
| EXTERNAL INTERFACES OVERVIEW .....                       | 84        |
| USING JDL FILES .....                                    | 84        |
| JDL Commands .....                                       | 84        |
| USING DYNAMIC DATA EXCHANGE .....                        | 86        |
| JDL Commands .....                                       | 86        |
| JDL PROPERTIES .....                                     | 87        |
| EXAMPLES OF USING JOB DEFINITION LANGUAGE COMMANDS ..... | 90        |
| <b>CHAPTER 18: SYSTEM OPTIONS .....</b>                  | <b>93</b> |
| General Options .....                                    | 93        |
| Fail-Save Mode and Distributed Service Properties .....  | 93        |
| Editor Options .....                                     | 94        |
| Log and Debug Options .....                              | 94        |
| Service Options for Windows NT .....                     | 95        |
| Service Options for Windows 95/98 .....                  | 95        |
| <b>CHAPTER 19: 24X7 JOB AUTOMATION LANGUAGE .....</b>    | <b>96</b> |
| OVERVIEW .....   | 96        |
| SYNTAX .....   | 96        |
| Variables .....  | 96        |
| Statements .....   | 97        |
| Statement continuation .....                             | 97        |
| Special ASCII characters .....                           | 98        |
| Off-line scripts .....                                   | 98        |
| Comments .....   | 98        |
| SCRIPT LIBRARY .....                                     | 98        |
| CONTROL-OF-FLOW STATEMENTS .....                         | 101       |
| Break .....  | 101       |
| Continue .....   | 101       |
| Dim .....  | 101       |
| Exit .....   | 102       |
| RaiseError .....   | 102       |
| ForNext .....  | 103       |
| GoTo .....   | 104       |
| If .....   | 104       |
| IfThen .....   | 105       |
| ChooseCase .....   | 106       |
| LoopWhile .....  | 107       |
| LoopUntil .....  | 108       |
| Set .....  | 109       |
| BITWISE STATEMENTS .....                                 | 109       |
| BitwiseAnd .....   | 109       |
| BitwiseClearBit .....                                    | 110       |
| BitwiseFlipBit .....                                     | 110       |
| BitwiseGetBit .....                                      | 111       |
| BitwiseOr .....  | 111       |
| BitwiseNot .....   | 112       |
| BitwiseSetBit .....                                      | 112       |
| BitwiseXor .....   | 113       |
| CLIPBOARD STATEMENTS .....                               | 113       |
| ClipboardGet .....                                       | 113       |
| ClipboardSet .....                                       | 114       |
| DATABASE STATEMENTS .....                                | 114       |
| DatabaseConnect .....                                    | 114       |
| DatabaseConnectEx .....                                  | 115       |
| DatabaseCopy .....                                       | 116       |
| DatabaseDelete .....                                     | 116       |
| DatabaseDescribe .....                                   | 117       |
| DatabaseDisconnect .....                                 | 117       |
| DatabaseExecute .....                                    | 117       |
| DatabaseExport .....                                     | 118       |

|                                |     |
|--------------------------------|-----|
| DatabaseGet .....              | 118 |
| DatabaseImport .....           | 119 |
| DatabaseInsert .....           | 119 |
| DatabasePaste .....            | 120 |
| DatabasePipe .....             | 120 |
| DatabaseRetrieve .....         | 122 |
| DatabaseRowCount .....         | 122 |
| DatabaseSave .....             | 123 |
| DatabaseSet .....              | 123 |
| DatabaseSetFilter .....        | 124 |
| DatabaseSetSort .....          | 125 |
| DatabaseSetSQLSelect .....     | 126 |
| DatabaseUpdate .....           | 126 |
| DATE AND TIME STATEMENTS ..... | 127 |
| Date .....                     | 127 |
| DateTime .....                 | 128 |
| DateAdd .....                  | 129 |
| DateDiff .....                 | 129 |
| DateTimeAdd .....              | 130 |
| DateTimeDiff .....             | 130 |
| DateTimePart .....             | 131 |
| DayName .....                  | 131 |
| DayNumber .....                | 132 |
| MakeDate .....                 | 132 |
| MakeDateTime .....             | 133 |
| MakeTime .....                 | 133 |
| Now .....                      | 134 |
| Timer .....                    | 134 |
| Time .....                     | 134 |
| TimeAdd .....                  | 135 |
| TimeDiff .....                 | 135 |
| Today .....                    | 136 |
| DDE STATEMENTS .....           | 136 |
| DDEClose .....                 | 136 |
| DDEExecute .....               | 137 |
| DDEGetData .....               | 137 |
| DDEOpen .....                  | 138 |
| DDESetData .....               | 138 |
| FILE STATEMENTS .....          | 139 |
| Dir .....                      | 139 |
| DirWaitForUpdate .....         | 140 |
| EOF .....                      | 140 |
| FileExists .....               | 141 |
| FileClose .....                | 141 |
| FileCopy .....                 | 142 |
| FileCopyEx .....               | 142 |
| FileTransfer .....             | 143 |
| FileMove .....                 | 143 |
| FileMoveEx .....               | 144 |
| FileDate .....                 | 144 |
| FileTime .....                 | 145 |
| FileDelete .....               | 145 |
| FileDeleteEx .....             | 146 |
| FileFindFirst .....            | 146 |
| FileFindNext .....             | 147 |
| FileSearchEx .....             | 147 |
| FileReplaceEx .....            | 148 |
| FileConvert .....              | 149 |
| FileGetAttr .....              | 151 |
| FileSetAttr .....              | 151 |
| FileGetPos .....               | 152 |
| FileSetPos .....               | 153 |
| FileOpen .....                 | 153 |
| FilePrint .....                | 154 |

|  |     |
|--|-----|
| FileRead.....  | 155 |
| FileReadAll.....                                     | 155 |
| FileReadLine.....                                    | 156 |
| FileRename.....                                      | 156 |
| FileSplitName.....                                   | 157 |
| FileSave.....  | 157 |
| FileSize.....  | 158 |
| FileWrite.....                                       | 158 |
| FileZip.....   | 159 |
| FileUnzip.....                                       | 160 |
| IniFileGetKey.....                                   | 160 |
| IniFileSetKey.....                                   | 161 |
| NotFileExists.....                                   | 162 |
| RemoteDir.....                                       | 162 |
| FILE REPLICATION AND SYNCHRONIZATION STATEMENTS..... | 163 |
| SyncFTPDDir.....                                     | 163 |
| SyncLocalDir.....                                    | 165 |
| SyncRemoteDir.....                                   | 166 |
| Dir.....   | 167 |
| FTPDDir.....   | 167 |
| RemoteDir.....                                       | 168 |
| FTP STATEMENTS.....                                  | 169 |
| FTPConfig.....                                       | 169 |
| FTPDDir.....   | 170 |
| FTPDeleteFile.....                                   | 170 |
| FTPFileExists.....                                   | 171 |
| FTPGetFile.....                                      | 172 |
| FTPPutFile.....                                      | 173 |
| FTPRenameFile.....                                   | 173 |
| JOB MANAGEMENT STATEMENTS.....                       | 174 |
| JobCreate.....                                       | 174 |
| JobDelete.....                                       | 175 |
| JobDescribe.....                                     | 175 |
| JobEnable.....                                       | 176 |
| JobGetStatus.....                                    | 177 |
| JobModify.....                                       | 178 |
| JobRun.....  | 178 |
| JobRemoteRun.....                                    | 179 |
| INTERNET STATEMENTS.....                             | 181 |
| Ping.....  | 181 |
| PingPort.....  | 182 |
| WebConfig.....                                       | 183 |
| WebGetFile.....                                      | 184 |
| WebPostData.....                                     | 185 |
| WebGetPageHTML.....                                  | 185 |
| WebOpenPage.....                                     | 186 |
| LOGICAL STATEMENTS.....                              | 187 |
| And.....   | 187 |
| IsDate.....  | 187 |
| IsDateBetween.....                                   | 188 |
| IsEqual.....   | 188 |
| IsGreater.....                                       | 189 |
| IsGreaterOrEqual.....                                | 189 |
| IsHoliday.....                                       | 190 |
| IsLess.....  | 190 |
| IsLessOrEqual.....                                   | 191 |
| IsNumber.....  | 191 |
| IsTime.....  | 192 |
| IsTimeBetween.....                                   | 192 |
| IsWeekday.....                                       | 192 |
| IsWeekend.....                                       | 193 |
| Not.....   | 193 |
| NotEqual.....  | 194 |

## Table of Contents

|                                |     |
|--------------------------------|-----|
| Or .....                       | 194 |
| MAIL STATEMENTS .....          | 195 |
| MailSend .....                 | 195 |
| MailSendWithAttachment .....   | 196 |
| MISCELLANEOUS STATEMENTS ..... | 197 |
| AgentTest .....                | 197 |
| Call .....                     | 197 |
| CD .....                       | 198 |
| DiskGetFreeSpace .....         | 199 |
| DiskGetFreeSpaceEx .....       | 199 |
| LogAddMessage .....            | 200 |
| LogAddMessageEx .....          | 201 |
| LogSearch .....                | 201 |
| LogSearchEx .....              | 202 |
| LogWaitForUpdate .....         | 203 |
| MemoryGetFree .....            | 204 |
| MessageBox .....               | 204 |
| Reboot .....                   | 205 |
| Yield .....                    | 205 |
| NUMERIC STATEMENTS .....       | 205 |
| Add .....                      | 205 |
| Ceiling .....                  | 206 |
| Divide .....                   | 206 |
| Floor .....                    | 207 |
| Max .....                      | 207 |
| Min .....                      | 207 |
| Mod .....                      | 208 |
| Multiply .....                 | 208 |
| Power .....                    | 209 |
| Round .....                    | 209 |
| Subtract .....                 | 210 |
| PRINT STATEMENTS .....         | 210 |
| FilePrint .....                | 210 |
| Print .....                    | 211 |
| PrinterGetDefault .....        | 211 |
| PrinterPurgeAllJobs .....      | 212 |
| PrinterSetDefault .....        | 212 |
| PROCESS STATEMENTS .....       | 213 |
| IsTaskRunning .....            | 213 |
| ProcessGetID .....             | 213 |
| ProcessGetHandle .....         | 214 |
| ProcessKill .....              | 215 |
| ProcessList .....              | 215 |
| ProcessGetWindow .....         | 216 |
| Run .....                      | 216 |
| RunAndWait .....               | 219 |
| RunWithInput .....             | 221 |
| ProcessGetExitCode .....       | 224 |
| SendKeys .....                 | 225 |
| Wait .....                     | 227 |
| WindowGetProcess .....         | 227 |
| RAS STATEMENTS .....           | 228 |
| RASDial .....                  | 228 |
| RASGetStatus .....             | 229 |
| RASHangUp .....                | 229 |
| REGISTRY STATEMENTS .....      | 230 |
| RegistryGetKey .....           | 230 |
| RegistrySetKey .....           | 231 |
| SERVICE STATEMENTS .....       | 232 |
| ServiceContinue .....          | 232 |
| ServiceGetStatus .....         | 232 |
| ServicePause .....             | 233 |

|  |            |
|--|------------|
| ServiceStart .....   | 234        |
| ServiceStop .....  | 234        |
| STRING STATEMENTS .....  | 235        |
| Asc .....  | 235        |
| Char .....   | 235        |
| Concat .....   | 236        |
| ConcatEx .....   | 236        |
| Fill .....   | 237        |
| Format .....   | 237        |
| GetToken .....   | 238        |
| InStr .....  | 238        |
| Left .....   | 239        |
| Length .....   | 239        |
| Lower .....  | 240        |
| LTrim .....  | 240        |
| Match .....  | 241        |
| Mid .....  | 241        |
| Number .....   | 242        |
| Pos .....  | 242        |
| Replace .....  | 243        |
| Reverse .....  | 244        |
| Right .....  | 244        |
| RTrim .....  | 244        |
| Space .....  | 245        |
| String .....   | 245        |
| Trim .....   | 246        |
| Upper .....  | 246        |
| TELNET STATEMENTS .....  | 247        |
| TelnetClose .....  | 247        |
| TelnetOpen .....   | 247        |
| TelnetSend .....   | 248        |
| TelnetReceive .....  | 249        |
| TelnetConfig .....   | 249        |
| WINDOW STATEMENTS .....  | 250        |
| WindowActivate .....   | 250        |
| WindowClickButton .....  | 251        |
| WindowClose .....  | 251        |
| WindowFind .....   | 252        |
| WindowGetActive .....  | 252        |
| WindowGetChild .....   | 253        |
| WindowGetFirst .....   | 253        |
| WindowGetLast .....  | 254        |
| WindowGetNext .....  | 255        |
| WindowGetParent .....  | 255        |
| WindowGetPrevious .....  | 256        |
| WindowGetProcess .....   | 256        |
| WindowGetTitle .....   | 257        |
| WindowWaitClose .....  | 257        |
| WindowWaitOpen .....   | 258        |
| WindowPostMessage .....  | 259        |
| WindowSendMessage .....  | 260        |
| <b>FUNCTIONS IN DATABASE FILTER AND SORT EXPRESSIONS .....</b> | <b>261</b> |
| DATA TYPE CHECKING FNCTIONS .....                              | 261        |
| IsDate .....   | 261        |
| IsNull .....   | 262        |
| IsNumber .....   | 262        |
| IsTime .....   | 263        |
| DATE AND TIME FUNCTIONS .....                                  | 263        |
| Date .....   | 263        |
| DateTime .....   | 264        |
| Day .....  | 264        |
| DayName .....  | 265        |
| DayNumber .....  | 265        |

|                               |     |
|-------------------------------|-----|
| DaysAfter .....               | 265 |
| Hour .....                    | 266 |
| Minute .....                  | 266 |
| Month .....                   | 266 |
| Now .....                     | 267 |
| RelativeDate .....            | 267 |
| RelativeTime .....            | 267 |
| Second .....                  | 268 |
| SecondsAfter .....            | 268 |
| Time .....                    | 268 |
| Today .....                   | 269 |
| Year .....                    | 269 |
| MISCELLANEOUS FUNCTIONS ..... | 269 |
| Case .....                    | 270 |
| GetRow .....                  | 271 |
| If .....                      | 271 |
| IsRowModified .....           | 271 |
| IsRowNew .....                | 272 |
| ProfileInt .....              | 272 |
| ProfileString .....           | 273 |
| RGB .....                     | 273 |
| NUMERIC FUNCTIONS .....       | 274 |
| Abs .....                     | 274 |
| Ceiling .....                 | 275 |
| Cos .....                     | 275 |
| Exp .....                     | 275 |
| Fact .....                    | 276 |
| Int .....                     | 276 |
| Integer .....                 | 276 |
| Log .....                     | 277 |
| LogTen .....                  | 277 |
| Long .....                    | 277 |
| Mod .....                     | 278 |
| Number .....                  | 278 |
| Pi .....                      | 278 |
| Rand .....                    | 279 |
| Round .....                   | 279 |
| Sign .....                    | 280 |
| Sin .....                     | 280 |
| Sqrt .....                    | 280 |
| Tan .....                     | 281 |
| Truncate .....                | 281 |
| STRING FUNCTIONS .....        | 281 |
| Asc .....                     | 282 |
| Char .....                    | 282 |
| Fill .....                    | 282 |
| Left .....                    | 283 |
| LeftTrim .....                | 283 |
| Len .....                     | 284 |
| Lower .....                   | 284 |
| Match .....                   | 284 |
| Metacharacters .....          | 285 |
| Sample patterns .....         | 286 |
| Mid .....                     | 286 |
| Pos .....                     | 287 |
| Replace .....                 | 287 |
| Right .....                   | 288 |
| RightTrim .....               | 288 |
| Space .....                   | 289 |
| String .....                  | 289 |
| Format Symbols .....          | 290 |
| Trim .....                    | 291 |
| Upper .....                   | 292 |
| WordCap .....                 | 292 |

---

|   |            |
|---|------------|
| OPERATORS.....  | 292        |
| <b>CHAPTER 20: SCRIPT DEBUGGER .....</b>                              | <b>295</b> |
| <b>CHAPTER 21: TESTING AND DEBUGGING.....</b>                         | <b>297</b> |
| <b>CHAPTER 22: JAL EXAMPLES.....</b>                                  | <b>298</b> |
| Running job between 9:00 AM and 5:00 PM every workday .....           | 298        |
| Redirecting program output to a file .....                            | 299        |
| Converting comma-separated data file to tab-separated file .....      | 299        |
| Using FTP commands.....   | 300        |
| Scheduling a job not to run on particular days (exception days) ..... | 300        |
| Converting dates to file names.....                                   | 301        |
| Scanning program log file for errors .....                            | 302        |
| Processing files using file mask .....                                | 302        |
| Collecting Web server performance statistics .....                    | 303        |
| Monitoring file-server free space .....                               | 304        |
| Using Windows messages .....  | 304        |
| Unattended server reboot on demand.....                               | 305        |
| Restoring network connection.....                                     | 305        |
| Watching for file changes.....  | 306        |
| Paging/notifying system administrators.....                           | 306        |
| Monitoring database free space .....                                  | 307        |
| Starting NT service, executing job, stopping NT service .....         | 309        |
| Loading data into database .....                                      | 310        |
| Establishing dial-up connection, avoiding line drops .....            | 311        |
| Verifying overnight data feed .....                                   | 311        |
| <b>CHAPTER 23: TECHNICAL SUPPORT .....</b>                            | <b>313</b> |
| <b>CHAPTER 24: FREQUENTLY ASKED QUESTIONS (FAQ) .....</b>             | <b>314</b> |
| <b>INDEX .....</b>  | <b>317</b> |

# Chapter 1: Getting Started

## Overview

## Preface

This manual describes all features of the 24x7 Scheduler, an advanced job scheduling and automation system for Microsoft Windows 95/98/NT/2000 platforms. Information in this manual applies to the 24x7 Scheduler v2.4.0 running on all supported operating systems. This manual contains information for both beginning and experienced users of the 24x7 Scheduler. Both the print and the on-line documentation assume that you have a working knowledge of standard Windows mouse and keyboard actions and understand Windows basic concepts. This manual is provided so that the reader can understand how 24x7 Scheduler functions. It also contains information on the following topics:

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

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

## Introduction

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

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

The 24x7 Scheduler enables you to:


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

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

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

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

On Windows NT 24x7 Scheduler provides an option to install 24x7 Windows NT service. This allows you to start 24x7 automatically when Windows NT starts regardless of user logon. If setup under system account the 24x7 service will run even when the user is logged off.

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

### Running on network

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

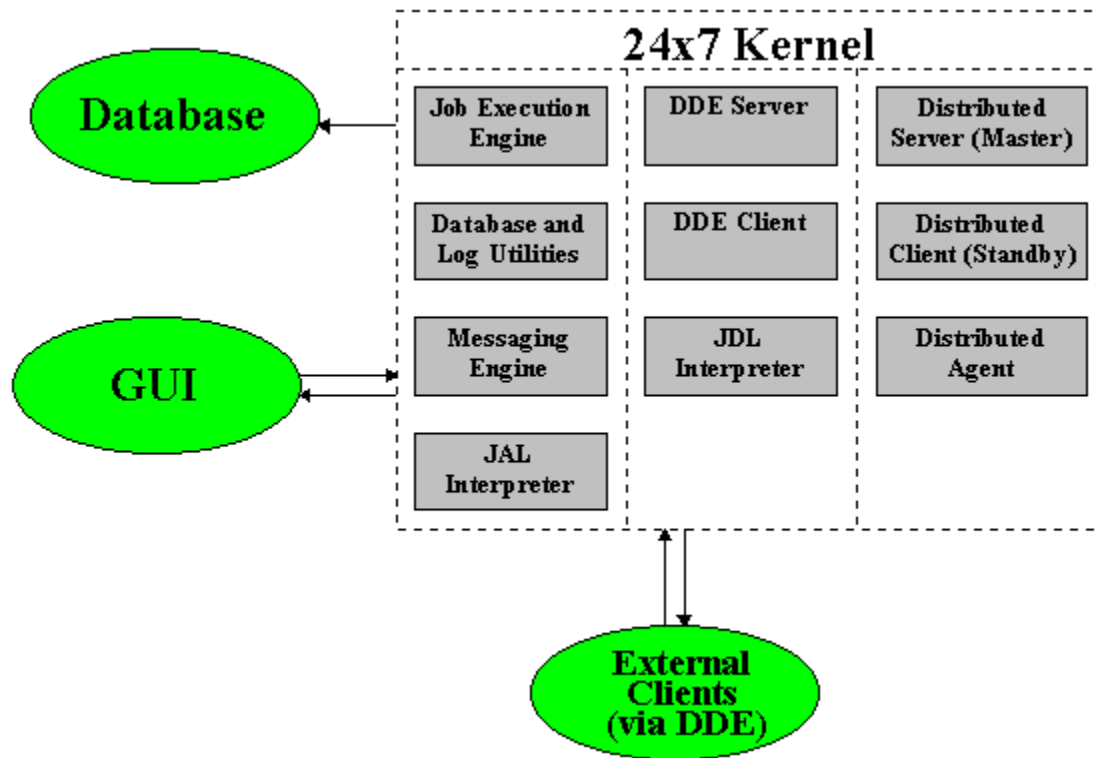
### Security issues

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

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

## 24x7 Scheduler Architecture

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



## Trademarks

24x7 Scheduler, ORACLE Monitor are trademarks of SoftTree Technologies, Inc. Windows 95, Windows 98, Windows NT are registered trademarks of Microsoft Corporation. All other trademarks appearing in this help file are trademarks of their respective owners. All rights reserved.

## See also:


- Job Explorer
- Job Properties Wizard

## To Start 24x7 Scheduler Each Time Windows Starts

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

## Running 24x7 Scheduler as a Windows NT Service

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

- The 24x7 Scheduler can start automatically whenever the computer is started and runs continuously in the background, regardless of whether a user is logged on.
- Programs run by the 24x7 schedule service inherit the security attributes of the scheduler. If the 24x7 schedule service is set to log on using local system account, the 24x7 schedule service acts as if it is part of the operating system. Therefore, all started programs get the same security permissions as the operating system. The 24x7 schedule service can be also set to using any valid Windows NT user account, so started programs will inherit the security profile of that user.
- Not all processes, applications, documents, and .bat files can be started from the 24x7 Scheduler. In order to run an interactive application, the application needs to be started with the local system account, and the "Allow service to interact with desktop" check box in Services Control Panel Applet must be checked. If the 24x7 schedule service is set to log on as the local system, scheduled jobs cannot be validated on the network so they cannot access any network resources. If you start the 24x7 schedule service with a user account, jobs can be validated on the network, but they can't have any user interface, because only the local system has sufficient privileges to allow a service to start a program on the interactive desktop. Unless 16-bit applications have access to the interactive desktop, they most likely not start as they run under Ntvdn. Therefore, you can have network access or interactivity with the system (needed by Ntvdn), but not both.  
 **All these limitations are by Windows NT design, for security purposes. You do not want a regular user to be able to schedule a job that would run when the administrator is logged on, and use the administrator's credentials.**
- When the 24x7 Scheduler runs as a service, you cannot do much with it other than stop it and start it. Instead, you use the standalone version of the 24x7 Scheduler to modify job database and customize 24x7 Scheduler options.



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

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

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

### See also:

System Requirements  
Installation and Uninstallation  
Service Options

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

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

- The 24x7 Scheduler can start automatically whenever the computer is started and run regardless of whether a user is logged on.
- A user can stop the 24x7 Scheduler service by right-clicking on the 24x7 Scheduler icon in the system tray, then selecting **Exit** command from the popup menu.

- When the 24x7 Scheduler runs as a service with "survive log off" option enabled, it can run other applications in the event of a user log-off. This can be done using jobs scheduled "on user log-off" . Such started applications run regardless of whether a user is logged on.



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

**See also:**

System Requirements  
Installation and Uninstallation  
Service Options

## Job Explorer

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

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

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

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

### Explorer "Tree View"

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

Collapsed folders represented by a closed folder icon

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

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

### Property Pages

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

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

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

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

## Status Bar

The status bar is a horizontal area that appears on the bottom of Job Explorer window. The status bar provides information about the current state of what you are viewing in the window and any other contextual information.



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

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

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

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

## System Tray Icon

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

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



### Tips:

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

### See also:

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

## Job Types

There are three types of jobs:

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

## Program Files and Documents

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



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

## Database Commands

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



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

## Job Automation Scripts

This type includes jobs that can execute user-defined scripts. The 24x7 Scheduler parses the script, checks syntax errors, then interprets and executes the script line by line. See Job Automation Language Help for details.

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

## File Types

To create a file type:

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

To modify the settings belonging to an existing file type:

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

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

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

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

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



**Tip:**

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

## Drag and Drop Interface

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

If you want to move a job to another folder:

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

When moving all jobs from one folder to another folder:

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

**See also:**

Job Explorer  
Support for Windows Explorer Drag and Drop

## Support for Windows Explorer Drag and Drop

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

You can add a file to the loaded schedule database by following these steps:

- 1 Start Windows Explorer or File Manager
- 2 While holding down the left mouse button, drag (move) the mouse pointer to the Job Explorer.
- 3 Release the mouse button to drop the file.

**See also:**

Job Explorer  
Support for Windows Explorer Drag and Drop

## Semaphore Files

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

There are two kinds of the semaphore files:

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

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

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

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

### See also:

- Job Execution Properties
- About Job Interdependencies
- Dependencies Editor Interface

## Execution Logs

### About main job execution log

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

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

### About NT system event log

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

To turn on/off logging in the Windows NT Application Log:

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

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

### About Status Report

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

To enable/disable Status Report updating:

- 1 Select the **Tools** command then select **Options**.
- 2 Check or uncheck **Generate Status Report (HTML)** option on the **General** tab.

### About other log files

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

To turn on/off tracing:

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



#### Tip:

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

#### See also:

Logging Job Execution  
Log Viewer  
Trace Features  
Job Execution Statistics

## Changes to the System Time

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

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

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

## Working with Job Database

The 24x7 Scheduler stores information on scheduled jobs in the **Job Database**. The **Job Database** consists of a single file with the default name SCHEDULE.DAT. By default, this file is located in the 24x7 Scheduler installation directory. You should not attempt to edit job database files directly. If you have installed the 24x7 Scheduler in more

than one directory or on two or more computers and you wish to copy information on scheduled programs between them, you can copy that file to replicate the job database. The 24x7 Scheduler also stores some configuration information in the System Registry under the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\SoftTree Technologies, Inc.\24x7`. This information includes all 24x7 Scheduler configuration options available through the Tools/Options menu, definitions of database profiles, and definitions of remote agent profiles. In case of system recovery you can try modifying this information directly, but it is highly recommended to use 24x7 Scheduler GUI.

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

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

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

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

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

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

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



**Tip:**

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

## Sending and Receiving E-mail Messages

### Overview

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



#### Important Notes:

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

## About Lotus Notes interface

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

- Notes e-mail interface library
- Notes extension manager

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



#### Warning:

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

## Maintenance for Holidays, Database Profiles, and Remote Agents

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

The following information is shared:

- Holiday list,
- Database Profile definitions,
- Remote Agent definitions.

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

#### See also:

Holiday List  
Database Profiles

## Remote Agent Profiles

## Chapter 2: Installation and Uninstallation

### Installation

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

- Typical,
- Compact,
- Custom.

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

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

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

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

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

### Windows NT specifics

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

Optionally, the 24x7 Scheduler can be setup to run as a Windows NT system service.

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

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

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

Q124184 - Service Running as System Account Fails Accessing Network.

Q132679 - Local System Account and Null Sessions in Windows NT.

Q152451 - Applications Run from the Schedule Service Fail to Print.

To enable or disable the Windows NT service, carry out the following:

- 1 Select the **Tools/Options** menu. The **Options** dialog will appear.
- 2 Select **Service** tab page.
- 3 Check "Run as a Windows NT service" .
- 4 Specify the desired settings for the 24x7 Scheduler Windows NT service.
- 5 Click the **OK** button.
- 6 Restart your computer.



You must restart your computer before new service related settings will take effect.

### Windows 95/98 specifics

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

To enable or disable the Windows 95/98 service application:

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

### Installing Remote Agents

There is no difference between installing a regular 24x7 Scheduler and a 24x7 Remote Agent. However, you must manually replicate Database Profiles and Holiday List settings in order to synchronize Remote Agent settings with the Master Scheduler. See Remote Agents topic for information on configuring Remote Agents.

### Automating the Installation Process

If you plan to install the 24x7 Scheduler using the same installation settings on a large number of computers, you can record the installation process on one computer and then play it back on to the others. This allows you to run the Setup program on the other computers automatically, without user input ("silent" installation).

A silent installation does not prompt the user for input. A silent installation receives its user input from the InstallShield Silent Response File (.ISS file).

To automate the installation process:

- 1 Start the setup program using the -r command-line option (e.g. SETUP -r).
- 2 Go through the installation process, selecting the options that you want to use for the automated installation.
- 3 After the installation process is complete, you will have a file called SETUP.ISS in your Windows (or WinNT) directory. Copy this file to the directory on your network with the rest of the setup files.
- 4 To run the automated setup on another computer, run the setup program with the command-line option -s (e.g. SETUP -s).
- 5 The setup program will now run silently and without user interaction.

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

To copy configuration information:


- 1 Install the 24x7 Scheduler on the first computer, recording the installation as described above.
- 2 Configure the 24x7 Scheduler using the **Tools/Options** menu.
- 3 Using the Registry Editor (REGEDT32.EXE), export the HKEY\_LOCAL\_MACHINE\SOFTWARE\SoftTree Technologies, Inc.\24x7\ registry key to a file with the extension .REG.
- 4 Copy this .REG file to the directory that contains the 24x7 Scheduler setup files.
- 5 Use the Registry Editor on the destination computer to import registry keys from the .REG file.
- 6 Restart the 24x7 Scheduler on the destination computer.

### Uninstallation

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

To uninstall the 24x7 Scheduler:

- 1 Click the Windows **Start** button.

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

**See also:**

System Requirements

## System Requirements

### Minimum Hardware Requirements

Intel-based machine running one of the following operating system:

- Windows NT 4.0 (server or workstation)
  - Windows 95
  - Windows 98
- 1 At least 16 MB RAM
  - 2 6.5 MB disk space
  - 3 VGA monitor

### Recommended Configuration

Pentium class CPU 150 MHz or better

- 1 32 MB RAM or better
- 2 SVGA 256-color or better monitor

### Fail-over Mode

- 1 Pentium class CPU 150 MHz or better
- 2 At least 32 MB RAM. For each Standby Scheduler add 2MB on the server machine
- 3 VGA monitor

### Software Requirements

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

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

**See also:**

Installation and Uninstallation

## Chapter 3: Database Interfaces

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

### Database Profiles Dialog Box

#### Description

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

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

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

#### New

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

#### Edit

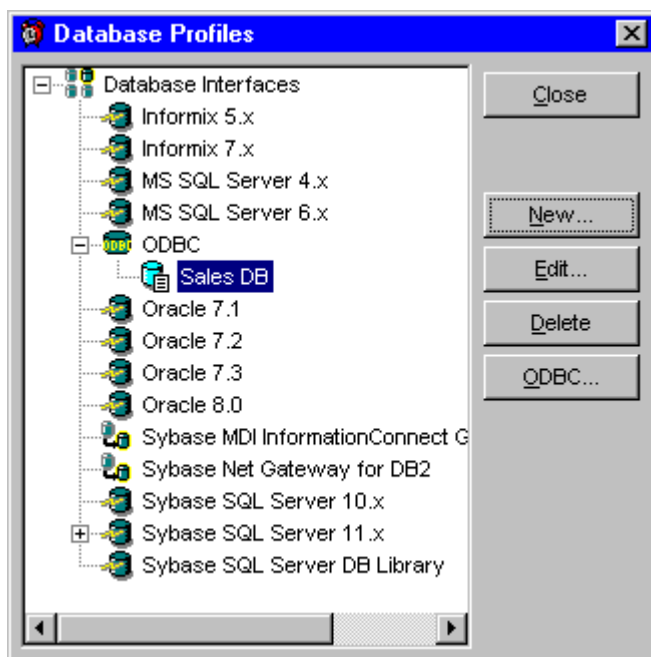
This will display the Database Profile Setup dialog box for the selected profile. You will be able to use this dialog to edit the profile. Select a database profile name and click **Edit**. This will display the Database Profile Setup dialog box for that profile.

#### Delete

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

#### ODBC

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



**See also:**

Database Jobs  
ODBC Interface  
Database Profiles

## Database Profiles

### Database Profile Setup

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

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

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

### Testing the connection to your database

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



#### Important Notes:

- You must have database client software installed on your computer before you can establish a database connection.
- In some databases, such as SQL Server, you must turn on the AutoCommit in order to run stored procedures executing SQL DDL statements. DDL statements include: CREATE INDEX, SELECT INTO temporary table, DROP TABLE and so on.

**See also:**

- Database Jobs
- Database Interfaces
- ODBC Interface
- Testing Connection to Your Database

## ODBC Interface

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

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

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

**See also:**

- Database Jobs
- Database Interfaces
- Database Profiles

## Chapter 4: Scheduling Jobs

### Job Wizard

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

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

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

**Tips:**

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

**See also:**

Using SQL and JAL Editors

## Job Properties

### General Properties

#### Job ID

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

#### Job Name

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

#### Description

The program provides a facility for you to enter in a job description, comments, and instructions. For your convenience, Job Explorer shows a few lines of the job description when you rest the mouse pointer over the job. Double-click the job in Job Explorer to start the Job Wizard. This will enable you to view and modify the full text if necessary.

#### Status

This term refers to job status. The status can be one of the following:

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

#### See also:

- Job Execution Properties
- Job Schedule and Triggers
- Notification Events and Actions
- Semaphore Files
- Macro-parameters
- Remote Jobs

## Job Execution Properties

#### Job Type

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

- **Program** – used to execute a program or a document file
- **Database command** – used execute SQL script
- **24x7 script** – used to execute JAL script

#### Program Name

The program name indicates the full or partial path and filename of the module to be run.

The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer. If the path or filename includes spaces you must use double quotes as module name delimiters. For example: "C:\Program Files\Daily Jobs\DataLoad.bat."

If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:

- 1 The directory from which the 24x7 Scheduler loaded.
- 2 The current directory.
- 3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.
- 4 The 16-bit Windows system directory. The name of this directory is SYSTEM.
- 5 The Windows directory. The name of this directory is WINDOWS or WINNT.
- 6 The directories listed in the PATH environment variable.

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

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

You can use **macro-parameters** inside the **program name** and command line parameters string to pass dynamic information (such as the current month) on to the scheduled program.

### Start In

The working directory is the directory in which the program executable finds the associated files it needs to run the program. Most programs do not require an entry in this field. Occasionally however, a program will not run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document.

You can use macro-parameters inside the **Start in** string to use dynamic information (such as the current month) as the start directory for the scheduled program.

### Window Type

This property tells the 24x7 Scheduler how the program (or document) needs to be started - maximized, minimized, normal, or hidden.

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

### Host (Remote Agent)

This specifies the name of the remote agent running on the host computer that will run the job.

### Backup Host (Backup Remote Agent)

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

### Restart Windows

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

### Asynchronous Process

For program and document file jobs:

24x7 Scheduler starts the specified program by taking advantage of Windows multi-tasking features. Otherwise, when a synchronous process is requested, 24x7 Scheduler starts the specified program (i.e. process) and enters an efficient wait state until this process finishes or until the time-out interval elapses. In the latter case, 24x7 Scheduler forcibly terminates the process. See also **Timeout** property.

New process inherits the Environment block from 24x7.

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

For JAL scripts and SQL database jobs:

The 24x7 Scheduler runs the job in the background using a separate thread, taking advantage of Windows multi-tasking features. Otherwise when a synchronous process requested, 24x7 Scheduler will execute the job using its main thread. This ensures that other jobs will not start until the current job processing is finished.

### Detached Job

Enabling this property is the same as running job manually from the command line using **24x7 /JOB** command. When this option is enabled, 24x7 Schedule spawns another instance that runs the job. Depending on the job value of **Asynchronous** property, the main instance either wait or don't wait for the job to complete.

24x7 takes advantage of Windows multitasking features that allow multiple instances of the same program to run concurrently.

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

**Caution:** You should enable Detached Process property only for jobs that affect or can potentially affect 24x7 Scheduler stability, that is, if the job crashes or cause some memory leak, it does not affect the main instance that will continue to run other jobs even after the detached job has crashed. Note that detached processes while running take some additional system resources.

### Timeout

Timeout defines the time interval, in minutes that the job is allowed to run before being terminated. Once time-out occurs, 24x7 Scheduler will terminate the process. If zero is specified, the interval never occurs, therefore the process can run on to infinity. This means that 24x7 Scheduler will continue with other job processing only after this job finishes. This property makes sense for synchronous processes (program executions) only.

### Send Keystroke

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

### Keystroke

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

### Init. Timeout

This term refers to the time that the 24x7 Scheduler will wait before sending a keystroke to the program or document it has started.

### Delay

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

**Caution:** When scheduling daily tasks, make sure that the start time + maximum delay falls on the same day as the start time. Otherwise, a late job may be skipped because of the day shift.

### Skip Job if Delay is Over <n> minutes

This instructs the 24x7 Scheduler to skip the late job where the delay is longer than the specified **Delay** interval. You use the **Delay** property to specify the maximum allowed delay (see above).

### Script

This term refers to the JAL script text that the 24x7 Scheduler executes. This parameter applies to the "24x7 Script" job type.

### Profile

The term “Profile” refers to the database profile name that describes the connection parameters for the database job. The specified profile must exist at the time that the job is being run. This parameter applies to the “Database command” job type.

### SQL

This refers to the SQL script text that the 24x7 Scheduler executes. This parameter applies to the “Database command” job type.

### Logging Enabled

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

### Ignore Errors

This term instructs the 24x7 Scheduler to continue running a job regardless of job run-time errors.

Examples:

A JAL script will continue to run even if the “division by zero” error occurred.

A job notification action will be attempted even if the database job itself failed while executing SQL script.

A semaphore file assigned to the job will be deleted (using the “delete after run” rule) even if the launched program terminated abnormally.

### Retry After Error

This term instructs the 24x7 Scheduler to retry the job if it fails. The job will be restarted after the specified **Retry Interval**. You should use the **Number of Retries** property to specify how many times you want the 24x7 Scheduler to retry a failing job before it marks that job as unsuccessful.

### Retry Interval

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

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

### Number of Retries

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


### See also:

- General Properties
- Job Schedule and Triggers
- Notification Events and Actions
- Semaphore Files
- Macro-parameters
- Remote Jobs

## Job Schedule and Triggers

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

- **Time watch** – a time condition tests for a specified time.
- **File watch** – a file condition tests for the presence of a specified file or group of files.

- **Process watch** – a process condition tests for the presence or absence of other programs running at the same time.
- **E-mail watch** – an e-mail condition tests for the presence of a specified e-mail message.
- **User watch** – a user condition tests for a user activity (you can control the user activity threshold via the settings on the currently selected Screen Saver; click the button  to open the display settings dialog).
- **System watch** – a system condition tests for a user log off or system shutdown

**User-defined conditions:** You can also check your own conditions using Job Automation Language commands in jobs of 24x7 Script type.

The following parameters are used to specify job conditions:

### Repeat Every

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

### Start Date

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

### Start Time

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

### End Date

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

### End Time

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

### Daily and Weekly Schedule

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

### Monthly Schedule

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

### Skip Job On Holiday

This parameter instructs the 24x7 Scheduler to skip the job on the scheduled day if that day is a holiday.

### Slide Job On Holiday

This parameter instructs the 24x7 Scheduler to skip the job on the scheduled day if that day is a holiday and then run the job on the next non-holiday.

### Semaphore File

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

### Semaphore Process

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

### Polling Interval

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

**Caution:** The 24x7 Scheduler can check for a file or group of files every second. However, frequent checking may cause high CPU utilization and heavy network traffic when verifying network files.

### Delete Semaphore File Rule

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

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

This parameter makes sense only for the “file watch” jobs.

### Account

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

### Password

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

### Message Text

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

### Subject

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

### Save Attachments

This instructs the 24x7 Scheduler to save attachments found in the e-mail message that triggered the job. This parameter is only valid for “e-mail watch” jobs.

**See also:**

- General Properties
- Job Execution Properties
- Notification Events and Actions
- Semaphore Files
- Macro-parameters
- Remote Jobs

## Notification Events and Actions

### Job Notification Options

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

This option may be used to:

- Notify system administrators and operators of events. This is especially convenient for controlling critical processes and may be used to page operations personnel in the event of a job failure. This enables operations personnel to respond rapidly to and correct any production issues.
- Create conditional e-mail reminders.
- Create semaphore files used for job interdependencies so that depended jobs can start as soon as they detect this semaphore file(s). This allows linking multiple jobs in one logical batch process as well as invoking jobs on demand such as “fix it” jobs. The 24x7 Scheduler allows very flexible job interdependencies. It even allows you to link jobs on multiple local and remote computers provided the 24x7 schedule service is running at that particular computer.

### Notification Events

The following notification events are supported:

- **Start** – occurs immediately after a job has started successfully.
- **Finish** – occurs immediately after a job has finished successfully.
- **Not found** – occurs when the specified program file or document has not been found resulting in a job not starting.
- **Error** – occurs when a job fails.

You can specify more than one event to be processed. However, you cannot specify different notification actions for different events of the same job. Alternatively, you can, with JAL script, create customized error handlers that will give you greater control and allow different notification processing for different events.

### Notification Actions

The following actions can be performed on the notification events:

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

The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. Should an error occur and logging is switched on, the error is automatically written into the log file. For more information on error types and the error log, see Logging Job Execution topic.

For this action type you must specify:

- 1 E-mail profile (account) to use for sending the notification message. For example: *Exchange Settings*  
The actual value may be different for different e-mail interfaces. For MAPI interface you should use name of the MAPI profile that you use when logging-on to the e-mail system. For Lotus Notes you should use the

- name of the user (or ID) used when logging-on to Lotus Notes. For SMTP interface you should use your e-mail address.
  - 2 E-mail password.
  - 3 E-mail recipient. Use comma (,) to separate names of recipients.
- **Execute SQL command(s)** – Executes one or more specified SQL commands using the specified database profile.  
For this action type you must specify:
    - 1 Name of database profile to be used when executing the notification message.
    - 2 SQL script to send to the back-end database.
  - **Create semaphore file(s)** – Creates one or more specified semaphore files on the specified local or network drives. The file contents depend on the triggered notification event and can be one of the following:
    - 1 "Job <name> started."
    - 2 "Job <name> finished."
    - 3 "Job <name> execution error: <error description>."
    - 4 "Job <name> execution error: Program not found."
- The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging-on is enabled, the error is automatically written in the log file. For more information on error types and error log, see Logging Job Execution topic.
- For this action type you must specify:
- 1 Full name of the file to be created, including path to the file. Use comma (,) to separate multiple files.

**See also:**

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

## Macro-parameters

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

- Semaphore file names
- Program names and directories
- SQL commands
- SQL notification messages (messages sent by executing one or more database commands)
- Job automation scripts

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

**Parameter Substitution**

The 24x7 Scheduler always applies macro-parameters before checking the job conditions and running the job. Some macro-parameters consist of two parts: the parameter identifier and the format mask. You must use double quotes around masks in order for the parameter to be interpreted correctly. The following tables list the supported parameters and masks.

| Identifier | Meaning   |
|------------|---|
| @W         | An integer (1-7) representing the current day of the week. Sunday is day 1, Monday is day 2, and so on. |

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

### Format Masks

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

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

### Two-digit years

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

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

### Examples:

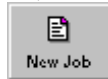
The following table shows how the date Friday, Jan. 30, 1998, displays when different format masks are applied:

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

## Adding New Job

### To schedule a new task

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



toolbar button

- 3 Follow the instructions in the Job Wizard.
- 4 You must save changes in the job database in order to apply them to the active job pool.



### Notes

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

### See also:

Working with Job Database

## Deleting Job

### To delete a scheduled job

- 1 In the Job Explorer, select the job want to delete.
- 2 Select **File** menu then select **Delete** command. Alternatively you can click the toolbar button
- 3 You must save changes in the job database in order to apply them to the active job pool.



### Tips:

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

### See also:

Stop Running Job

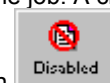
Working with Job Database

## Disabling/Enabling Job

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

### To temporarily disable a scheduled job

- 1 In the Job Explorer, select the job you want to delete.
- 2 Select **File** menu, then select **Disabled** command. The 24x7 Scheduler will disable the job. A checkmark will



- appear next to the **Disabled** menu item. Alternatively, you can click the toolbar button
- 3 You must save changes to the job database in order to apply them to the active job pool.

### To enable a previously disabled job

Repeat the steps described above.



#### Tips:

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

#### See also:

Stop Running Job  
Working with Job Database

## Modifying Job Definition and Schedule

### To rename a job

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

### To disable/enable a job

See Disabling/Enabling Job topic above for detailed instructions.

### To change all other job properties

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

## Moving Job to Another Folder

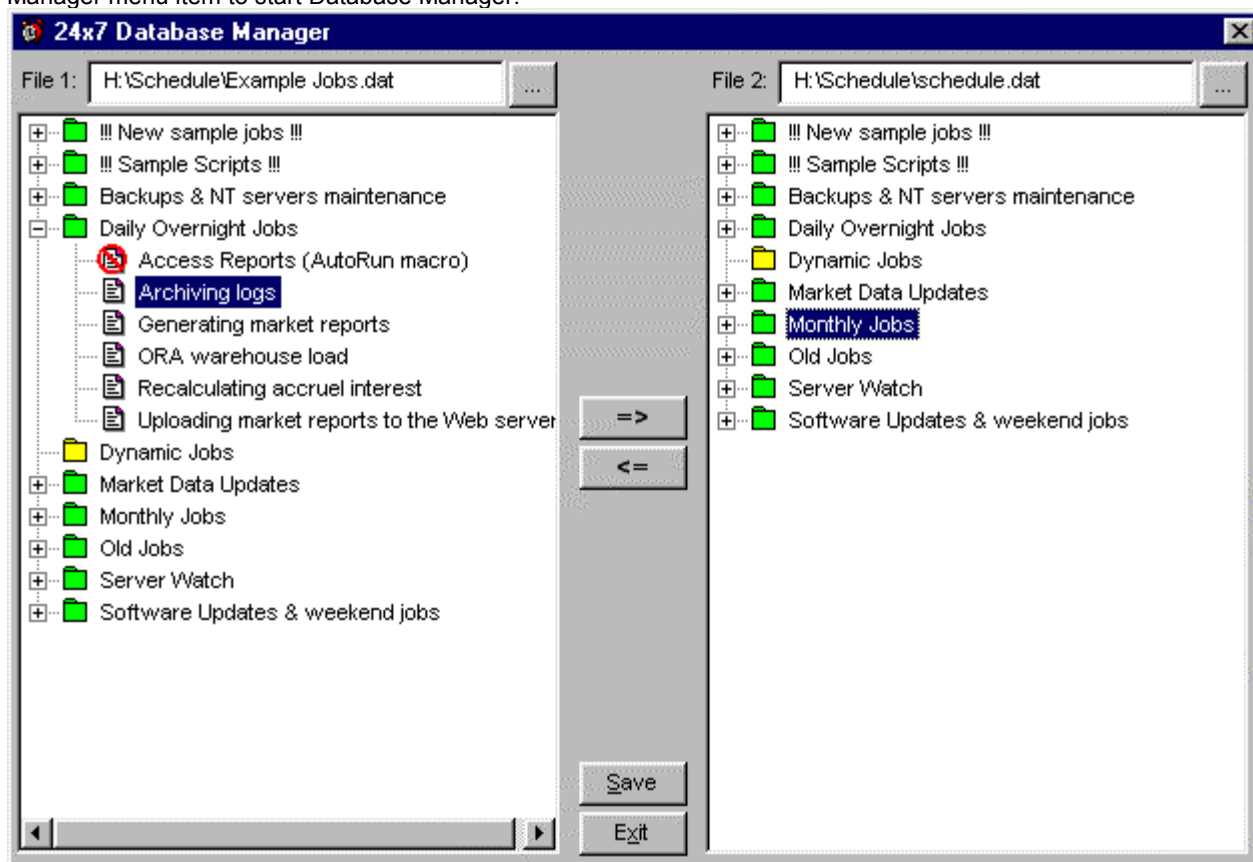
Use drag-and-drop to perform this operation. See Drag and Drop Interface topic for details.



#### See also:

Working with Job Database

## Copying Job to/from Another Database

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



1. Type in the **File 1** field the name of the first job database that you want to display in the left-hand part of the screen. Alternatively, you can press the  button to activate the **File Open** dialog box.
2. Type in the **File 2** field the name of the second job database that you want to display in the right-hand part of the screen. Alternatively, you can press the  button to activate the **File Open** dialog box.
3. Click on the source job that you want to copy then click on the destination folder in another database where you want to place the copied job. Select the desired source folder to copy all jobs from this folder.
4. Press the appropriate **Arrow Right/Left** button to perform the copy operation. You can also use drag-and-drop to achieve the same result.
5. When you have finished copying jobs, press the **Save** button to save changes. Click the **Exit** button to close Database Manager.



### Notes:

- If one of the modified databases was the same as the current database, you must reopen it by using **File/Open** command from the 24x7 Job Explorer menu.
- When copying a job, Database Manager copies all job properties with the exception of the job ID. A copied job is allocated the next job ID available in the destination database. That is why it is a good idea to use job names instead of job IDs when referring to a job from other jobs.

### See also:

Working with Job Database

## Protecting/Unprotecting Job

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

- assign a password to modify the job. This will prevent unauthorized users from modifying, deleting or disabling the job, but allows them to view job properties and scripts as well as execute the job.
- assign a password to view and modify the job. This allows others to execute the job but not to modify, delete or disable/enable the job.
- assign a password to execute, view, and modify the job. This does not allow others to execute, modify, delete or disable/enable the job.



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

### To protect/unprotect a job:

- 1 Select the desired job in **Job Explorer**.
- 1 Select **File/Protect Job** command. The job protection dialog box will appear.
- 2 Type your password then click the **OK** button.



### Notes:

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

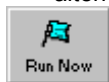
### See also:

Working with Job Database

## Testing Job Execution

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

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



### Important Note:

When an error occurs in the **Run Now** mode, 24x7 Scheduler will display modal message boxes that require input from the user. You should always close these messages as they can, potentially, prevent other jobs from starting in time.

Normally, the 24x7 Scheduler displays non-modal message boxes only. These non-modal message boxes will close by themselves after a few minutes. This allows 24x7 Scheduler to work unattended. You can also close these messages manually by clicking on the desired message box button.

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



**Tips:**

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

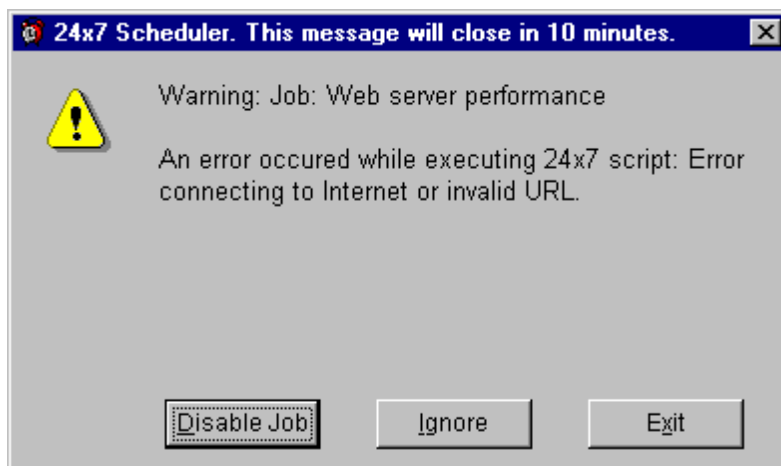
**See also:**

Disabling/Enabling Job

## Error Messages

When an error occurs in the **Run Now** mode, the 24x7 Scheduler will display modal message boxes that require input from the user. You should always close these messages as they can, potentially, prevent other jobs from starting in time.

Normally, the 24x7 Schedule displays non-modal message boxes only. These non-modal message boxes will close by themselves after a few minutes. This allows 24x7 Scheduler to work unattended. You can also close a non-modal message manually by clicking on the desired message box button. See error message box example below.



Button description:

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

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

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



**Tip:**

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

## Stopping Job Execution

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

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

**Note:**

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

## Creating Job Shortcut

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

You can start a single job by executing the following command from the DOS system prompt:

**24x7 /JOB <job id>**

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

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

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

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

**Notes:**

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

## Sending Keystrokes To Other Programs

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

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

| Key | Code  |
|-----|-------|
| Tab | {TAB} |

|                     |   |
|---------------------|---|
| Enter               | {ENTER} or {CR} or {RETURN}                                       |
| Esc                 | {ESCAPE} or {ESC}   |
| Backspace           | {BACKSPACE} or {BS} or {BKSP}                                     |
| Break               | {BREAK}   |
| Caps Lock           | {SCROLLLOCK} or {SCROLL LOCK}                                     |
| Caps Lock           | {CAPSLOCK} or {CAPS LOCK} or {CAP}                                |
| Del                 | {DELETE} or {DEL}   |
| Down Arrow          | {DOWN}  |
| End                 | {END}   |
| Help                | {HELP}  |
| Home                | {HOME}  |
| Ins                 | {INSERT}  |
| Left Arrow          | {LEFT}  |
| Num Lock            | {NUMLOCK} or {NUM LOCK}   |
| Page Down           | {PGDN} or {PAGE DOWN} or {PAGEDOWN}                               |
| Page Up             | {PGUP} or {PAGE UP} or {PAGEUP}                                   |
| Print Screen        | {PRTSC} or {PRINT} or {PRINT SCREEN}                              |
| Right Arrow         | {RIGHT}   |
| Up Arrow            | {UP}  |
| F1...F12            | {F1}...{F12}  |
| Ctrl                | {CTRL}  |
| Shift               | {SHIFT}   |
| Alt                 | {ALT}   |
| Numpad 0...Numpad 9 | {NUMPAD0}...{NUMPAD9} or {NUMPAD 0}...{NUMPAD 9}                  |
| Numpad +            | {NUMPAD+} or {NUMPAD +}   |
| Numpad -            | {NUMPAD-} or {NUMPAD -}   |
| Numpad *            | {NUMPAD*} or {NUMPAD *}   |
| Numpad /            | {NUMPAD/} or {NUMPAD /}   |
| Numpad .            | {NUMPAD.} or {NUMPAD .}   |
| Numpad Enter        | {NUMPADENTER} or {NUMPAD ENTER} or {NUMPAD RETURN} or {NUMPAD CR} |

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

- When sending a special key with a single character, precede the character code with one of the special keys: {CTRL} for Ctrl, {SHIFT} for Shift and {ALT} for Alt. For example: to send the key combination **Alt M** use **{ALT}m**
- When sending a special key with a group of characters, use parentheses to group the characters and precede the group code with one of the special keys: {CTRL} for Ctrl, {SHIFT} for Shift and {ALT} for Alt. For example: to send the key combination **Alt M** then **Alt G** then **Alt X** use **{ALT}(mgx)**

If you want to send braces and parentheses {, }, (, ) as a literal text, enclose the character in braces. For example, to send an open parenthesis use {{}.

**Note:**

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

**Example:**

123 A. {SHIFT}{:}{:}{ENTER}{ALT}{F4}N

The keystroke above is equivalent to pressing keys 1, 2, 3, space, A, period, :, ", :, ', Enter, Alt+F4, N

## Troubleshooting Job Execution

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

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

## Job Execution Statistics

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

**Sample entry from the statistics file**

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

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

## Disabling Timer

If you should have difficulties starting the 24x7 Scheduler or modifying job schedules, you can start it with the /NOTIMER option. To do this, run the 24x7 Scheduler using the following command line: **24x7 /NOTIMER**. This will disable the timer used by the Event Processor and as a result it will halt all job processing until the 24x7 Scheduler is restarted without the /NOTIMER option. All other 24x7 Scheduler features and functions will continue to operate.

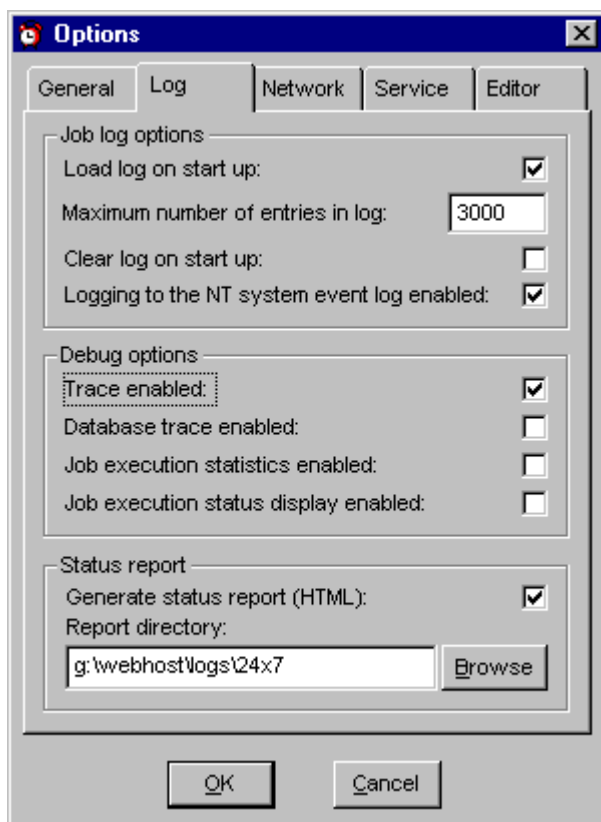
## Trace Features

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

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

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

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



To enable tracing for:

- 24x7 Master Scheduler session (on the Master Scheduler's computer)
- 24x7 Standby Scheduler session (on the Standby Scheduler's computer)
- 24x7 Remote Agent session (on Remote Agent's computer)

- JAL scripts execution tracing
- JDL script files execution tracing
- External interface session

you will need to do the following:

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

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

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

To enable database tracing:

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

To enable ODBC API Call tracing:

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

**See also:**

Database Profiles  
Remote Agent Profiles  
Troubleshooting Job Execution  
Logging Job Execution  
About Fail-over Mode

## Chapter 5: Job Interdependencies

### Overview

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

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

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

#### See also:


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

### Graphical Dependencies Editor

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

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

#### "Tree View"

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

Collapsed folders are represented by a closed folder icon .

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

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

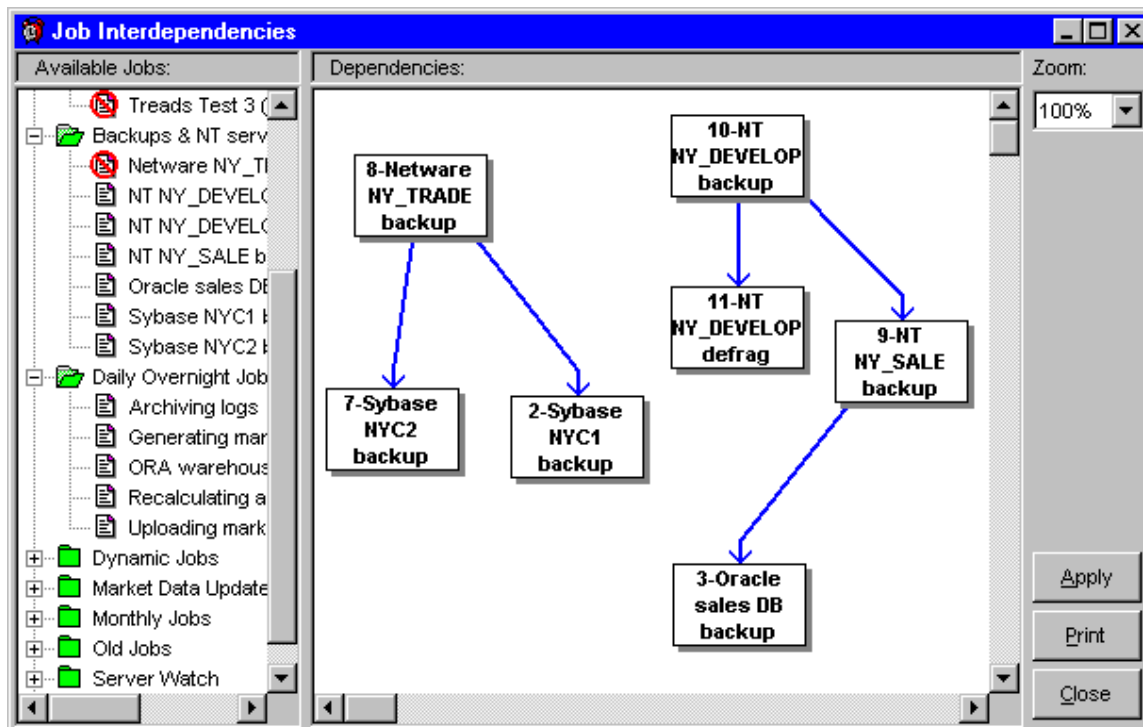
Navigating through the various folders and jobs is usually accomplished by clicking individual folders and jobs with the mouse. Click the plus sign (+) to expand a folder, or you can double-click on the folder itself. Click the minus sign (-) to collapse a folder, or you can double-click on the folder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

#### Dependencies Diagram

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

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

Use the zoom feature for customizing the dependencies view.



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



#### Tips:

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

#### See also:

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

## Adding New Job to Dependencies View

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

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

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

To add a job to the Dependencies View:

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

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

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

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

### See also:

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

## Deleting Job from Dependencies View

To delete a job from Dependencies View:

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

### See also:

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

## Arranging Jobs on Dependencies View

To arrange job object rectangles on the job Dependencies View:

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

### See also:

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

## Adding New Dependency

To add a new job dependency:

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

### See also:

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

## Deleting Dependency

To delete a job dependency:

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

### See also:

- About Job Interdependencies

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

## Printing Dependencies

To print the job dependencies diagram:

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

### See also:

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

## Using Zoom Tool

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



### Tip:

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

### See also:

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

## Chapter 6: Fail-over Mode

### About Fail-over Mode

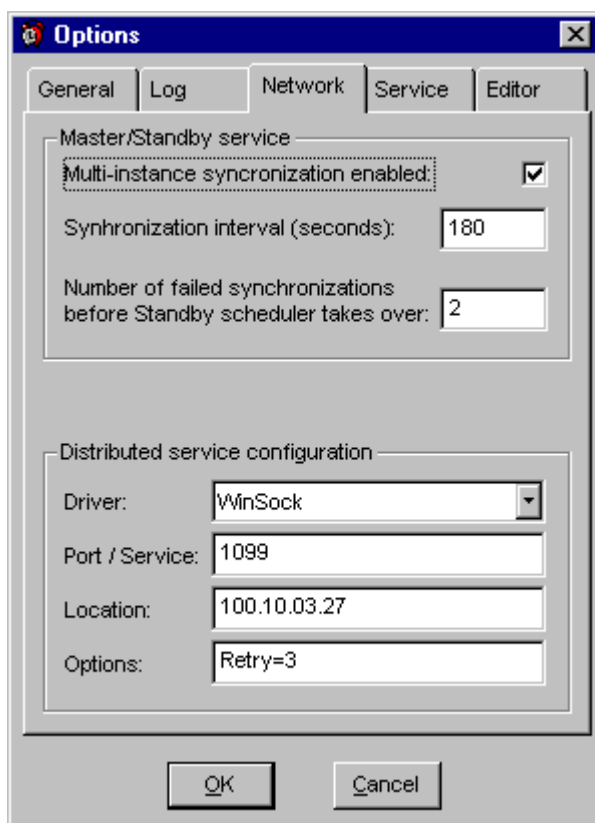
The "Fail-over Mode" is a "crash protection" mode in which the 24x7 Scheduler ensures high availability and error-recovery, reducing possible system downtime that can impede important processing functions, and even stop time-critical computing that is vital to your business.

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

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

To enable "Fail-over Mode" :

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

**Tips:**

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

**See also:**

Setting Connection Parameters  
 Testing Connection to Master Scheduler  
 Starting Master and Standby Schedulers

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

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

- WinSock
- NamedPipes
- Local

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

| Driver  | Parameter value  |
|---------|--|
| Winsock | Specify either of the following: <ul style="list-style-type: none"> <li>• The port number for the 24x7 Master Scheduler or Remote Agent (for example, 10099). Each server application requires a unique port number on the server machine. If you specify a port number, select a number that is greater than 4096 and less than 65536.</li> </ul> |

|            |   |
|------------|---|
|            | <ul style="list-style-type: none"> <li>The service name for the 24x7 Master Scheduler. The service name is an indirect reference to the 24x7 Scheduler's port number. The mapping of the service name to the port number is specified in the TCP/IP services file. Normally, this file (SERVICES.) is located in the Windows directory for Win95/98 and in the C:\WINNT\SYSTEM32\DRIVERS\ETC directory for WinNT. You may need to edit this file manually to add the 24x7 Scheduler Port or Service.</li> </ul> |
| NamedPipes | Specify the application portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. The Application must be unique for the Location you specify.  |
| Local      | This property is ignored.   |

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

| Driver     | Parameter value  |
|------------|--|
| Winsock    | Specify either of the following: <ul style="list-style-type: none"> <li>The IP address (for example, 199.99.99.91)</li> <li>The host name of the remote computer (network computer name in workgroup)</li> <li>LocalHost (This tells the 24x7 Standby Scheduler that the 24x7 Master Scheduler resides on the local machine.)</li> </ul> |
| NamedPipes | Specify the location portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. If no location is specified, a local pipe name is constructed using a dot (.) in the machine name portion.                              |
| Local      | This property is ignored.  |

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

|                              |   |
|------------------------------|---|
| <b>BufSize=n</b>             | Sets the connection buffer size to the value specified.   |
| <b>MaxListeningThreads=n</b> | Determines the maximum number of listening threads available on the Master Scheduler and Remote Agents.   |
| <b>MaxRetry=n</b>            | Specifies how many times the Standby Scheduler will try to connect when the Master Scheduler's listening port is busy. Applies to the <b>WinSock</b> driver only.   |
| <b>NoDelay=1</b>             | Specifies that each packet be sent without delay. Corresponds to the TCP_NODELAY option. Setting this option may degrade performance significantly. Do not use this option unless you thoroughly understand its implications. Applies to the <b>WinSock</b> driver only.  |
| <b>RawData=1</b>             | Specifies that raw data be passed over the network. By default, the <b>WinSock</b> driver obscures the data that is passed over the network. Setting this option to 1 overrides the default behavior. Both the Standby and Master Schedulers must have the same setting. If there is a discrepancy between the Standby and Master Scheduler's settings, the communication will fail. Setting this option to 1 may |

|  |  |
|--|--|
|  | improve performance slightly. Applies to the <b>WinSock</b> driver only. |
|--|--|

**See also:**

About Fail-over Mode  
Testing Connection to Master Scheduler  
Starting Master and Standby Schedulers

## Testing Connection to Master Scheduler

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

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

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

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

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

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

**Tips:**

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

**See also:**

About Fail-over Mode  
Setting Connection Parameters  
Starting Master and Standby Schedulers

## Starting Master and Standby Schedulers

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

**Connection troubleshooting:**

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



**Tip:**

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

**See also:**

- About Fail-over Mode
- Setting Connection Parameters
- Testing Connection to Master Scheduler

## Restricting Access to Master Scheduler

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

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

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

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



**Important Notes:**

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

**See also:**

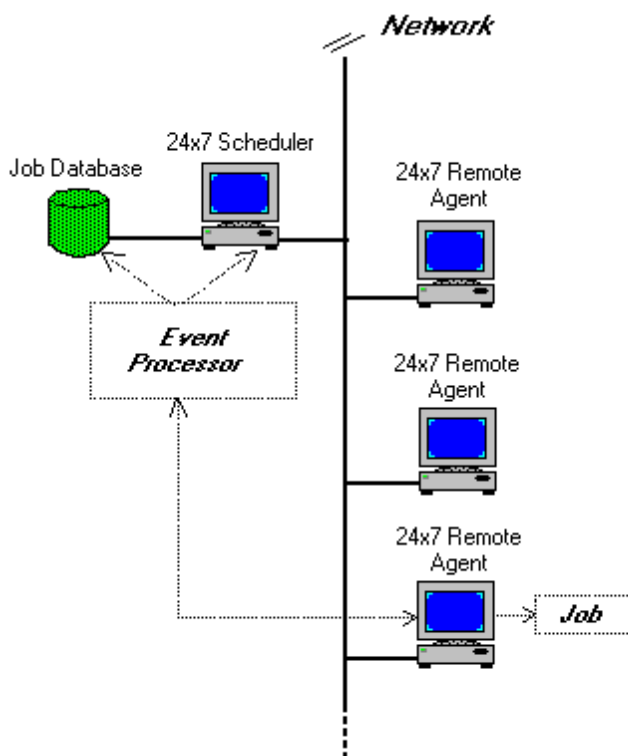
- About Fail-over Mode


## Chapter 7: Remote Agents

### About Remote Agents

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

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



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

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

The main 24x7 Scheduler (event processor) will attempt to connect to the 24x7 Remote Agent whose name is specified in the job properties. If the Remote Agent is not installed or it is not running on the selected computer, an error will occur and the job will fail. The error code and text will be written to the job log. If the connection succeeded, the main 24x7 Scheduler will submit the job definition to the Remote Agent. If this job has the **asynchronous** option turned off, the main 24x7 Scheduler will wait while the job is being executed by the Remote Agent on the remote computer. If the job has the **asynchronous** option turned on, the main 24x7 Scheduler will post the job for asynchronous execution on the remote computer and immediately after that, continue normal job processing.

#### See also:

Starting Remote Agent

## Starting Remote Agent

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

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

### To create a shortcut on the Desktop:

Alternatively, you may want to create a shortcut to start the 24x7 Remote Agent.

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

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



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

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

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

### See also:

About Remote Agents  
Example Remote Setup

## Restricting Access to Remote Agents

### To restrict access to a Remote Agent:

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

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

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



**Important Notes:**

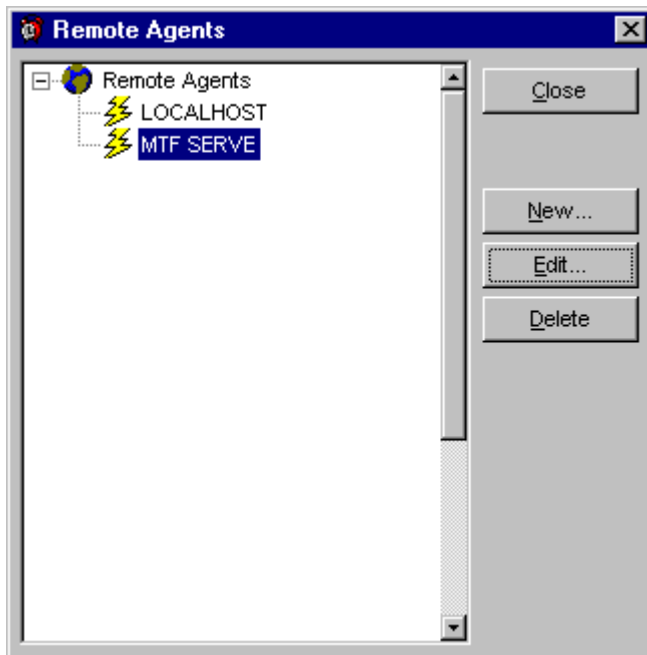
- Each installed copy of the 24x7 Scheduler must have a unique serial number. Please see license agreement for details.
- The list of allowed requestors RESTRICT.LST is also used to restrict 24x7 Standby Schedulers able to connect to the 24x7 Master Scheduler. Remember this when switching between Remote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

**See also:**

About Remote Agents

## Remote Agent Profiles

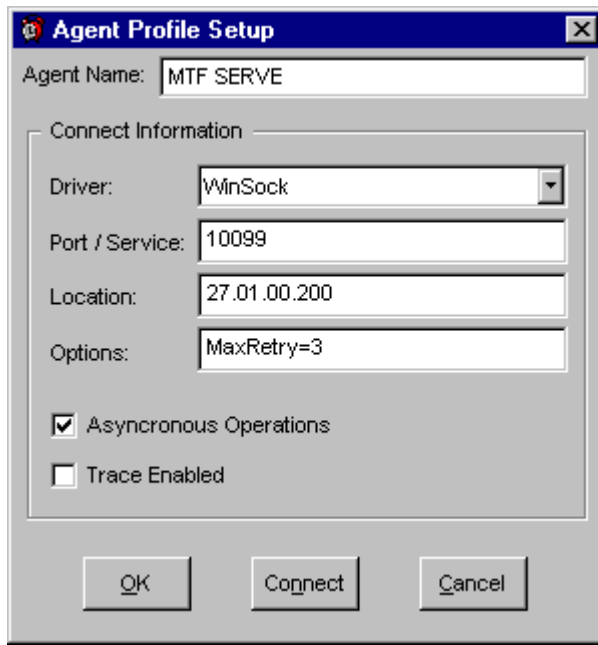
Before a job can be submitted to the 24x7 Remote Agent, the Remote Agent profile must be created on the main 24x7 Scheduler. The main 24x7 Scheduler uses profile information when connecting to the Remote Agent. Each agent requires a separate profile.



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

**To add a new profile:**

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



The image shows a Windows-style dialog box titled "Agent Profile Setup". It has a standard title bar with a close button (X). The dialog contains the following fields and controls:

- Agent Name:** A text box containing "MTF SERVE".
- Connect Information:** A section header followed by a group box containing:
  - Driver:** A dropdown menu showing "WinSock".
  - Port / Service:** A text box containing "10099".
  - Location:** A text box containing "27.01.00.200".
  - Options:** A text box containing "MaxRetry=3".
- Asynchronous Operations:** A checked checkbox.
- Trace Enabled:** An unchecked checkbox.
- Buttons:** At the bottom are three buttons: "OK", "Connect", and "Cancel".

**To modify an existing profile:**

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

**To delete a profile:**

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

**See also:**

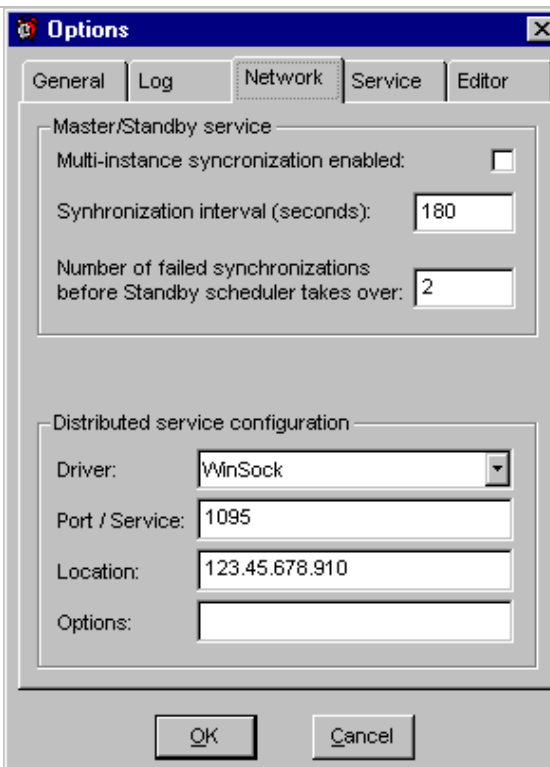
- About Remote Agents
- Setting Connection Parameters
- Example Remote Agent Setup

## Sample Remote Agent Setup

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

On the 24x7 Remote Agent computer:

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



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

On the main 24x7 Scheduler computer:

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



### See also:

Remote Agent Profile  
Connection Parameters

## Synchronous and Asynchronous Connections

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

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

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

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

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

**See also:**

- About Remote Agents
- Setting Connection Parameters
- Job Execution Properties

## Chapter 8: Database Jobs

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

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

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

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

## Preparing to use your database

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

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

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

## Installing the ODBC driver or native database driver

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

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

## Defining the ODBC data source

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

### From the 24x7 Scheduler

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

### From Windows Control Panel

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

### Completing the ODBC setup dialog box

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

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

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

## Creating database profiles

To create a new database profile:

- 1 Click **Tools** menu, then click **Database Profiles**.
- 2 Follow instructions described in the Database Profiles topic.

## Troubleshooting the database connection

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

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

### See also:

- Database Interfaces
- ODBC Interface
- Database Profiles
- Testing Connection to Your Database

## Chapter 9: Job Automation Scripts

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

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

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

**See also:**

- Job Automation Language Overview
- Job Automation Language Examples

## Chapter 10: Logging Job Execution

Logging job execution is optional, however, it is highly recommended for mission-critical jobs. The job log provides a complete audit trail for all job activities. You can use the information stored in the log to troubleshoot incorrect or “not on-time” job execution, and resolve scheduling conflicts between different jobs.

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

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

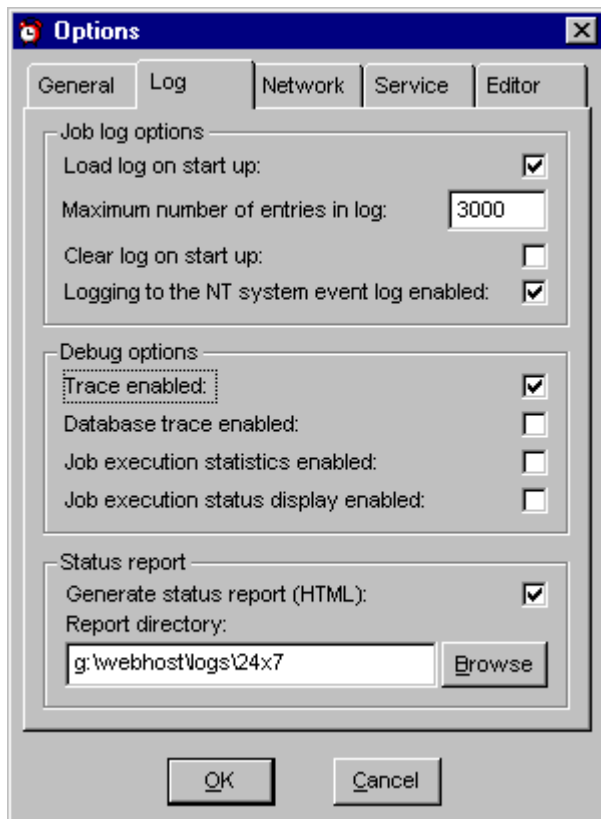
Contact SoftTree Technologies Technical Support for assistance If you are unable to resolve the error you are experiencing.

**See also:**

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

## Chapter 11: Status Report

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



To enable generation of the Status Reports:

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



### Tips:

- For the **Status report directory**, you can choose the directory on your Web server. In this case, all status report changes will be available immediately (real-time) to Internet users. Alternatively, you may want to setup a job that will upload created HTML files to the desired Web server. That job can be run every 30 minutes or at another specified interval providing near real-time reporting over the Internet.
- You should setup some security restrictions on your Web server if you do not want the report to be available to all Internet users.

### See also:

Using Web Browser to see Status Report  
Job Monitor

## Using Web Browser to See Status Report

To launch the default Web browser with the Status Report in it, click the **View** menu then select **Status Report**. If you have configured the 24x7 Scheduler to update Status Reports on your Web server, you will also be able to browse them on the Web server.

You can use the Web browser "find" feature to find the job you are looking for.

You can also use the Web browser to print the Status Report.

### **See also:**

Status Report

## Chapter 12: Exception Dates

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

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

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

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

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

### See also:

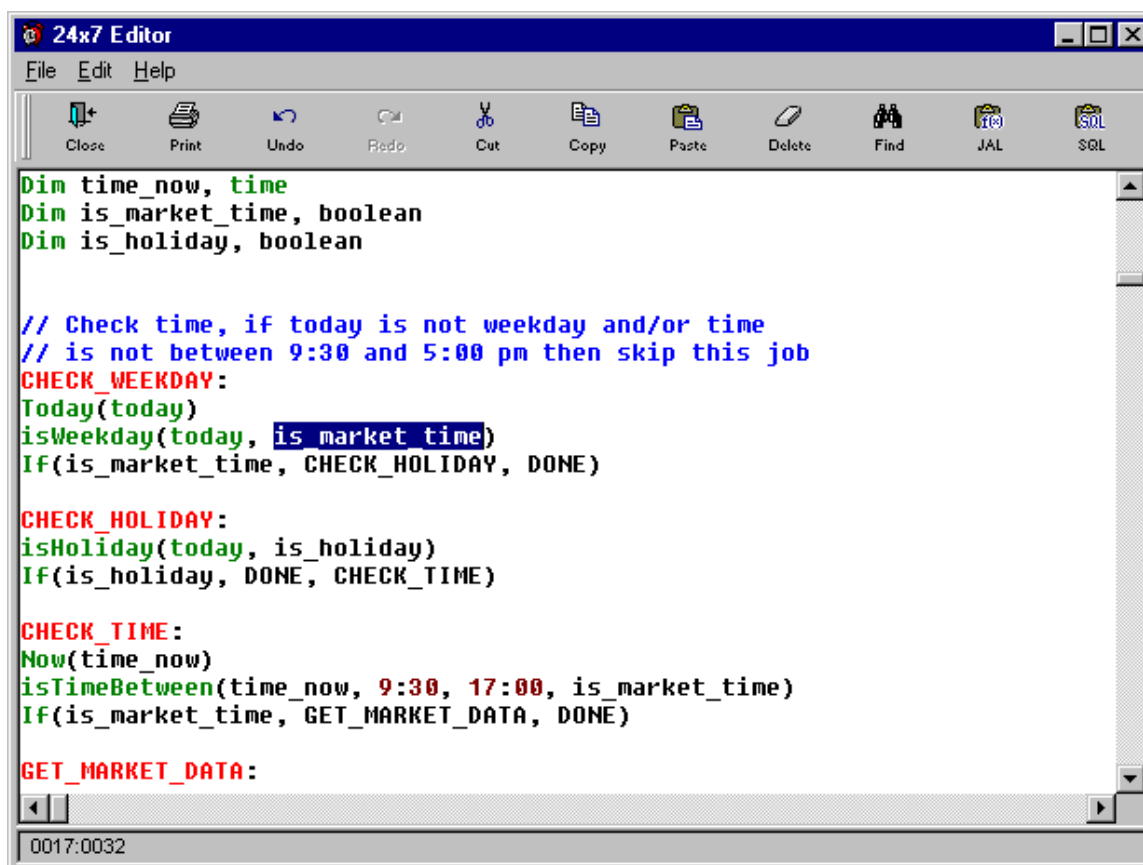
- Installation and Uninstallation
- Job Schedule

## Chapter 13: Editor

### Description

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

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



### Using the Editor

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

### Entering JAL and SQL scripts

You can enter a script in three ways:

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

### Searching in Scripts



- 1 Click the **Find** button on the Editor window toolbar or select **Edit/Find** command from the menu. The **Find** dialog box will open.

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

### Replacing in Scripts

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

### Go to Command

To go to a line:

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

To go to a previously set bookmark:

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

### Importing and Exporting Scripts

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

To export a script:

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

To import a script:

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

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

### Printing Scripts

To print a script:



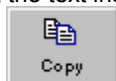
- 1 Click the **Print** button on the Editor window toolbar. The Print dialog box will open.
- 2 Select the **Printer** from the drop-down list. It should contain a list of local and network printers that you can access. If you do not see any listed, then your computer is not configured for any printers.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

### Copying, Pasting and Cutting Text

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

To copy and paste text:

- 1 Highlight the text inside the Editor window.



- 2 Click the **Copy** button on the Editor's toolbar. This action causes the selected text to be copied to the clipboard.

- 3 Place your cursor at the place where you want to paste the text. Click the **Paste** button. The text will be copied onto the script.



To cut and paste text:

- 1 Highlight the text inside the Editor window.



- 2 Click the **Cut** button on the Editor's toolbar. This action causes the selected text to be copied to the clipboard and removed from the script.

- 3 Place your cursor at the place where you want to paste the text. Click the **Paste** button. The text will be copied onto the script.



### Deleting the text

To delete the text:

- 1 Highlight the unwanted text inside the Editor window.



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

### Undo/Redo Changes

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

To undo a change:

- 1 Click the **Undo** button on the Editor's toolbar.

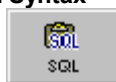


To repeat the last undone change:

- 1 Click the **Redo** button.



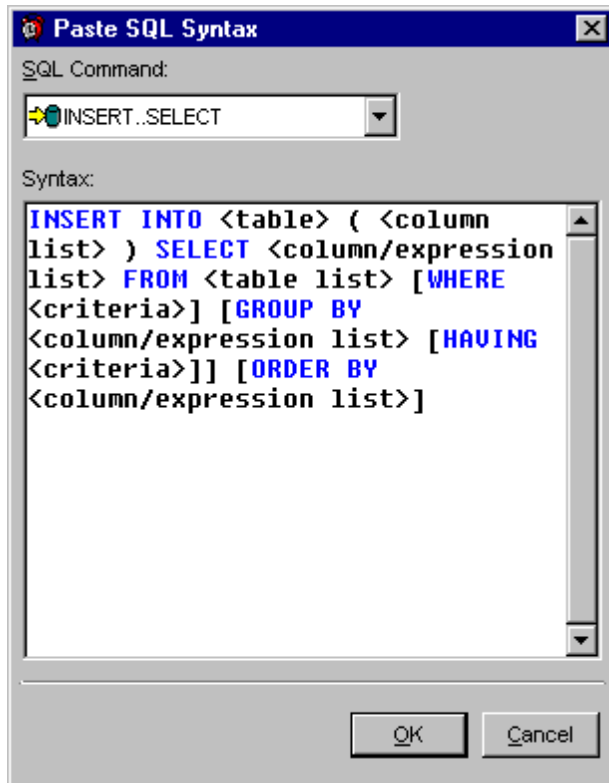
### Pasting SQL Syntax



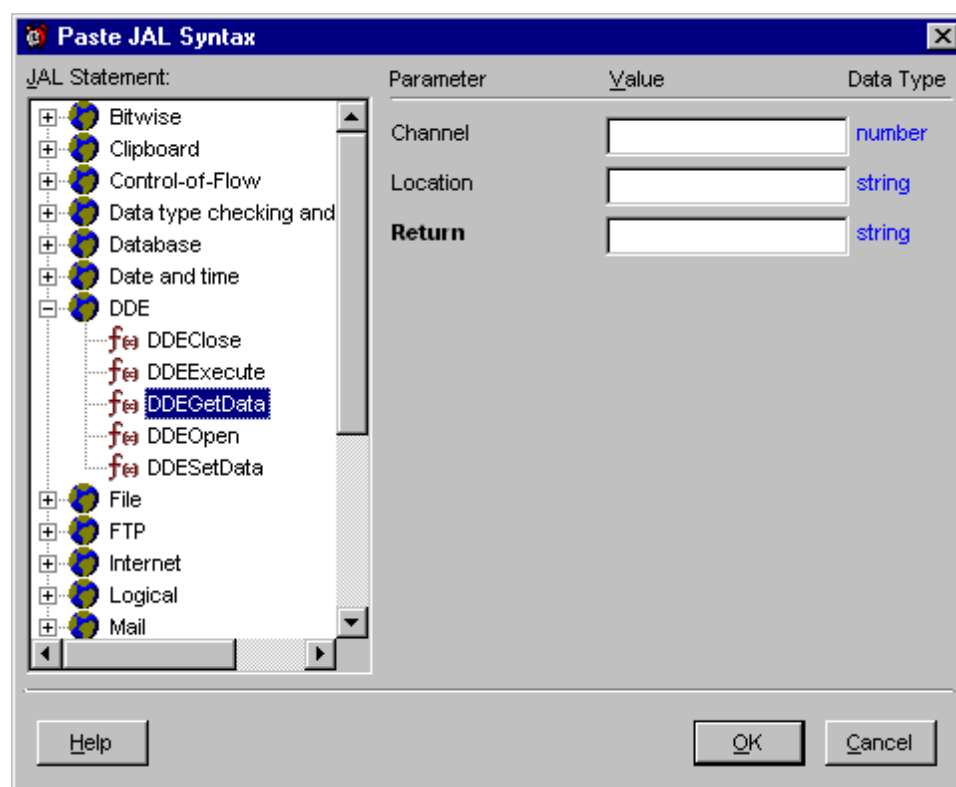
- 1 Click the **Paste SQL** button on the Editor's toolbar. The Paste SQL Syntax dialog box will open.
- 2 Select the command that you want to paste from the drop-down list of available SQL commands. The syntax will appear in the Syntax box.
- 3 Click the **OK** button. The 24x7 Scheduler will paste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

**Note:**

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

**Pasting JAL Syntax**

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



### Hot Keys

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

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

## Chapter 14: Log Viewer

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

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

The Log Viewer consists of the six following tab pages:

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

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

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

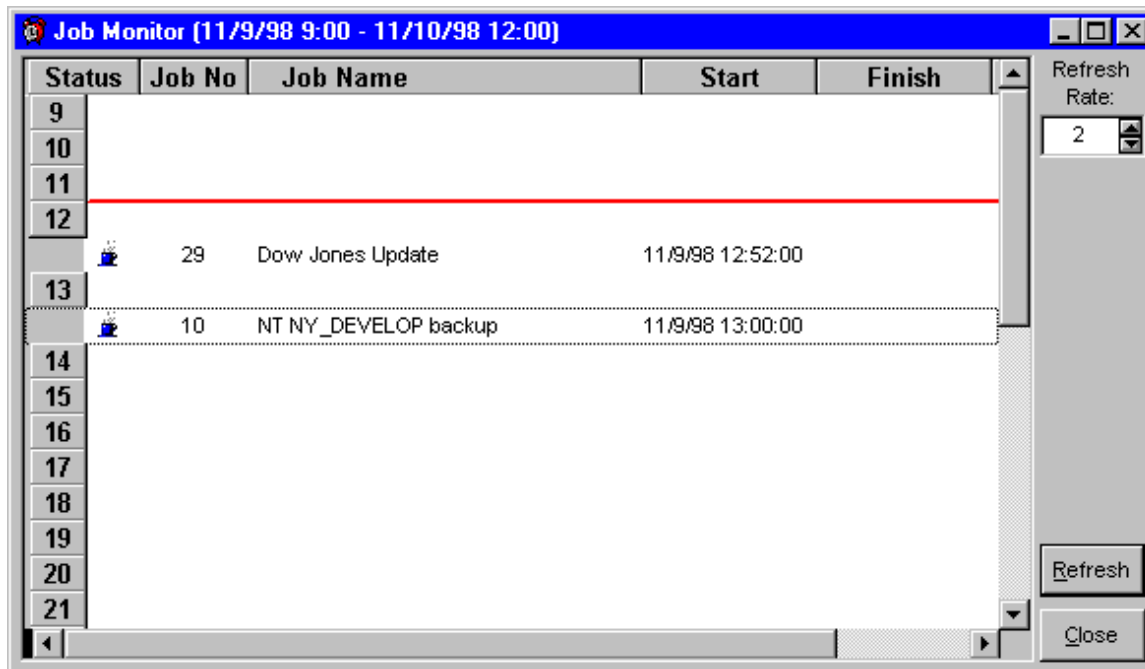


### Tips:

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

## Chapter 15: Job Monitoring

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



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

**Note:**

The Job Monitor cannot forecast jobs with non-time based schedules such as "file watch", "process watch", and "e-mail watch", as it does not know when the specified conditions will be met.

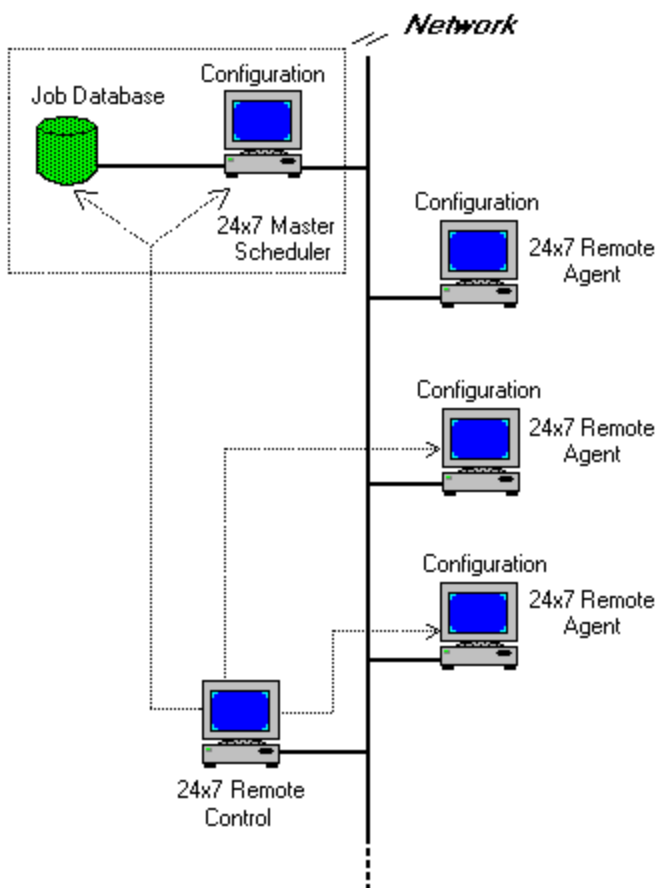
**See also:**

Status Report


## Chapter 16: 24x7 Remote Control

### About 24x7 Remote Control

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



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

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

24x7 Remote Control supports the following functions:

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

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

**See also:**

About Remote Agents  
Remote Agent Profiles  
Managing Remote Jobs and Configurations

## Managing Remote Jobs and Configurations

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

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

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

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

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

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



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

**See also:**

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

## Starting 24x7 Remote Control

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

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

### To create a shortcut on the Desktop:

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

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

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

### See also:

About 24x7 Remote Control

## Chapter 17: 24x7 Scheduler API - External Interfaces

### External Interfaces Overview

The 24x7 Scheduler provides four different APIs for programmatic job control:

- **DOS command prompt interface** – this interface is based on the simple Job Definition Language (JDL). You can manipulate job definitions via this interface using the **24x7** command executed from the DOS prompt. This command requires JDL file name passed as one of the command line parameters. A JDL file is a flat ASCII text file that can be created manually in any text editor or be generated dynamically using virtually any programming language. For details, see Using JDL Files topic. The DOS command interface is suitable for managing jobs 24x7 Scheduler running on the local computer.
- **DDE interface** – this interface is based on the standard Windows Dynamic Data Exchange (DDE) protocol. You can establish DDE links and call DDE functions from many programming environments such as MS Visual Basic, MS Access, C, PowerBuilder, Delphi, or any other language that can supports Windows DDE methods. For details, see Using DDE topic. The DDE interface is suitable for managing jobs on 24x7 Scheduler running on the local computer.
- **COM+ interface** (called **24x7 Remote Control COM+ interface**) – this interface can be used by any program that supports COM automation such as ASP, MS Visual Basic, MS Visual C++, MS Access, Delphi, PowerBuilder and many other. For details, see **24x7 Remote Control COM+ manual**. The COM+ interface is suitable for managing jobs and controlling 24x7 Scheduler running on the local or remote computer. This interface requires that the 24x7 Scheduler must be running in a server mode: either as a **Master Scheduler** or **Remote Agent**.
- **Java interface** (called **24x7 Remote Control Java interface**) – this interface can be used by any Java program. Such program can run as standalone program, as an applet in a Web browser or as a servlet in Web server. For details, see **24x7 Remote Control Java manual**. The Java interface is suitable for managing jobs and controlling 24x7 Scheduler running on the local or remote computer. This interface requires that the 24x7 Scheduler must be running in a server mode: either as a **Master Scheduler** or **Remote Agent**.

All external interfaces provide a way for programs to interact with the 24x7 Scheduler job engine in a consistent way.

### Using JDL Files

In addition to an advanced graphic user interface, the 24x7 Scheduler supports DOS command line interface for accessing and manipulating job properties. You can use the 24x7 command with the /SCRIPT option and JDL file name as the parameter, for example, **24x7 /SCRIPT SUBMIT.JDL**. The 24x7 Scheduler JDL files are regular text files that consist of JDL “commands”, job “property names” and “values”. A “property” must have an assigned “value” in the form **property=value**. A JDL file may also include comments. Comments may appear on a single line that starts with a semicolon (;). You can have as many comment lines as you want. Blank lines may be used to improve readability. They do not need to be preceded by a semicolon. Spaces and tabs can also be used to improve readability.



#### Notes:

- “Commands” and “property names” are case-insensitive; “values” are case-sensitive.
- You can shortcut command names to three or more first characters, for example, **Delete**, **DELETE**, and **DEL**.. All three variants will delete the specified job.

You can use the Windows text editor to create and modify JDL files.

### JDL Commands

The following JDL commands are supported:

- **ADD** – Adds a new job.  
Format: **ADD**  
**<property 1>=<value 1>**

```
<property 2>=<value 2>
<property 3>=<value 3>
...
<property n>=<value n>
```

Parameters:

**<property 1> .. <property n>** - valid property name. See JDL Properties topic for the complete list of property names.

**<value 1> .. /,value n>** - new value for the specified property.

You may specify as many properties as you want. Although specifying one or more properties is optional, it is highly recommended that you specify a unique name for each new job.



**Note:** A new job is created in the **Dynamic Jobs** folder. If this folder does not exist, it will be created automatically.

- **DELETE** – Deletes the specified job.  
Format: **DELETE <job>**  
Parameters:  
    **<job>** - valid job ID or job name.
- **DISABLE** – Disables the specified job.  
Format: **DISABLE <job>**  
Parameters:  
    **<job>** - valid job ID or job name.
- **ENABLE** – Enables the specified job.  
Format: **ENABLE <job>**  
Parameters:  
    **<job>** - valid job ID or job name.
- **GET** – This command is available only via DDE interface.
- **SAVE** – Saves all changes in the job database to the specified job.  
Format: **SAVE**  
Parameters:  
    This command has no parameters.
- **SET** – Alters the specified property of the specified job.  
Format: **SET <job> <property>=<value>**  
Parameters:  
    **<job>** - valid job ID or job name.  
    **<property>** - valid property name. See JDL Properties topic for complete list of property names.  
    **<value>** - new value for the specified job property.

The 24x7 Scheduler writes results of JDL file processing into JDL.LOG file in the 24x7 installation directory. You can use Windows Notepad or any other text viewer program to see contents of this file after processing. In addition, if the tracing features are enabled, the 24x7 Scheduler will log the internal JDL tracing information into the INTFACE.LOG file. You will be able to read this using the Log Viewer. This information can help when troubleshooting the JDL interface.

**See also:**

- JDL File Sample
- JDL Overview
- Using DDE
- Job Properties

## Using Dynamic Data Exchange

You can use standard Windows Dynamic Data Exchange (DDE) protocol for programmatic job control from other applications. DDE provides a link for two Windows applications to communicate. Using DDE, one application (the client application), can request information from, or send commands to, another application (the server application). The server application then processes the request from the client application. The server performs a task such as updating data, or returning requested information to the client, such as an element of data maintained by the server application.

The 24x7 Scheduler is designed to act as a DDE server, with the ability to process DDE requests and commands from client applications.

To use DDE based JDL interface, your application must act as a DDE client. The following steps are required for typical DDE communication:

- 1 The client application establishes a "cold" DDE link to the 24x7 Scheduler by calling the corresponding DDE function. This function name varies in different programming languages. For example, in Visual Basic for Applications - it is [DDEInitiate](#), in PowerBuilder - it is [OpenChannel](#). You would use "24x7 Scheduler" for the DDE name and "JDL" for the DDE topic parameters required for establishing a DDE link to the 24x7 Scheduler.
- 2 The client application executes one or more DDE requests.
- 3 The client application terminates the DDE link by calling the corresponding DDE function. This function name varies in different programming languages. For example, in Visual Basic - it is [DDETerminate](#), in PowerBuilder - it is [CloseChannel](#).

For additional information on DDE, consult Microsoft's DDE documentation.

## JDL Commands

The following JDL commands are supported:

- **ADD** – Adds a new job.  
Format: [ADD](#)  
Method: DDE command  
Parameters:  
    This command has no parameters.  
Return: None. You should immediately execute the **GET** command with the **NEW\_ID** parameter to get the new job ID. You can call the **SET** command later to set/modify properties of the new job by using the returned job ID as a reference.  
Visual Basic example:  
    [Application.DDEExecute ChannelNumber, "ADD" 'create new blank job](#)  
    [New\\_Job = Application.DDERequest ChannelNumber, "New\\_ID" 'retrieve ID of newly created job](#)



### Notes:

A new job is created in the **Dynamic Jobs** folder. If this folder does not exist, it will be created automatically.

- **DELETE** – Deletes the specified job.  
Format: [DELETE <job>](#)  
Method: DDE command  
Parameters:  
    <job> - valid job ID or job name.  
Visual Basic example:  
    [Application.DDEExecute ChannelNumber, "DELETE 5"delete job #5](#)
- **DISABLE** – Disables the specified job.  
Format: [DISABLE <job>](#)  
Method: DDE command  
Parameters:  
    <job> - valid job ID or job name.  
Visual Basic example:  
    [Application.DDEExecute ChannelNumber, "DISABLE 5"disable job #5](#)

- **ENABLE** – Enables the specified job.  
Format: `ENABLE <job>`  
Method: DDE command  
Parameters:  
    <job> - valid job ID or job name.  
Visual Basic example:  
    Application.DDEExecute ChannelNumber, "ENABLE 5"enable job #5
- **GET** – Retrieves the specified property of the specified job.  
Format: `GET <job><tab><property>`  
Method: DDE request data  
Parameters:  
    <job> - valid job ID or job name.  
    <tab> - tab character (ASCII code 9)  
    <property> - valid property name. See JDL Properties topic for complete list of property names.  
Return: value for the specified property  
Visual Basic example:  
    Start\_Time = Application.DDERequest ChannelNumber, "5"+ Chr(9) + "Start\_Time" 'retrieve start time for job #5
- **SAVE** –Saves all changes in the job database.  
Format: `SAVE`  
Method: DDE command  
Parameters:  
    This command has no parameters.  
Visual Basic example:  
    Application.DDEExecute ChannelNumber, "SAVE" 'save changes in the job database
- **SET** – Alters the specified property of the specified job.  
Format: `SET <job><tab><property> <value>`  
Method: DDE send data  
Parameters:  
    <job> - valid job ID or job name.  
    <tab> - tab character (ASCII code 9)  
    <property> - valid property name. See JDL Properties topic for complete list of property names.  
    <value> - new value for the specified job property.  
Visual Basic example:  
    Application.DDEPoke ChannelNumber, "5"+ Chr(9) + "Start\_Time" , "9:30"change start time for job #5

If tracing features are enabled, the 24x7 Scheduler will log the internal DDE tracing information into the INTFACE.LOG file. You can view this information using the Log Viewer. This information can help when troubleshooting the JDL interface



**Note:**

Before you establish a DDE link, make sure that the 24x7 Scheduler is running.

**See also:**

- JDL Interface Examples
- JDL Overview
- Using JDL Files
- Job Properties

## JDL Properties

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

| Name                 | Meaning   |
|----------------------|---|
| <b>ACCOUNT</b>       | E-mail Account such as user ID, profile, or e-mail address (e-mail watch job). The actual value may differ for different e-mail interfaces. For a MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to the Lotus Notes. For SMTP you should use your e-mail address.   |
| <b>AGENT</b>         | Same as Host (see Host description)   |
| <b>ASYNCH</b>        | Asynchronous Process, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>BACKUP_AGENT</b>  | Same as Backup Host (see Backup Host description)   |
| <b>BACKUP_HOST</b>   | Backup Remote Host (e.g. Backup Remote Agent name)  |
| <b>COMMAND</b>       | Program Command Line  |
| <b>DAY_NAME</b>      | Monthly Schedule Day Name, one of following: <b>Monday</b> , <b>Tuesday</b> , <b>Wednesday</b> , <b>Thursday</b> , <b>Friday</b> , <b>Saturday</b> , <b>Sunday</b> , <b>Weekend</b> , <b>Weekday</b>  |
| <b>DAY_NUMBER</b>    | Monthly Schedule Day Number, a number from 1 – 31 range   |
| <b>DELAY</b>         | Allowed Job Delay Interval (minutes)  |
| <b>DELETE_RULE</b>   | Delete Semaphore File Rule, one of the following: <b>D</b> , <b>A</b> , <b>B</b> (do not delete, delete after job run, delete before job run)   |
| <b>DESCRIPTION</b>   | Job Description   |
| <b>DISABLED</b>      | Job Disabled Status, one of the following: <b>Y</b> , <b>N</b> (yes, no)  |
| <b>DETACHED</b>      | Detached Job, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>END_DATE</b>      | Last Job Start Date   |
| <b>END_TIME</b>      | Last Job Start Date   |
| <b>FILE</b>          | Semaphore File Names(s) for file-watch jobs; Module Name for process-watch job  |
| <b>FOLDER</b>        | Job Folder ID. This is read-only property. It may not be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command   |
| <b>FOLDER_NAME</b>   | Job Folder Name. This is read-only property. It may not be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command   |
| <b>FRIDAY</b>        | Execute Job On Fridays, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>HOST</b>          | Remote Host (Remote Agent Name)   |
| <b>ID</b>            | Job ID, This is read-only property. It may not be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command with Job Name parameter.   |
| <b>IGNORE_ERRORS</b> | Ignore Errors, one of the following: <b>Y</b> , <b>N</b> (yes, no)  |
| <b>INIT_TIMEOUT</b>  | Initial Timeout before sending keystroke (seconds)  |
| <b>INTERVAL</b>      | Repeat Interval for Job having Schedule Type <b>T</b>   |
| <b>JOB_PASSWORD</b>  | <p>Job Protection State and Password. Sets or removes job protection state and password. This is a write-only property. It can be changed using <b>SET</b> command, but it cannot be retrieved using <b>GET</b> command. The value in this property must be specified in the following format :</p> <p>[old password][tab character][new password][tab character][protection state]</p> <p>If the job is not protected, the [old password] is ignored, otherwise a valid password must be specified in order to remove or change job password or protection state. If the protection exists and the new protection state is specified as an empty string the protection will be removed. The protection code must one of the following: <b>F</b>, <b>E</b>, <b>R</b>, an empty string (<b>F</b> – full protection; <b>E</b> – execute only; <b>R</b> – read only; an empty string indicates that a job is not protected).</p> |

|                          |   |
|--------------------------|---|
| <b>JOB_TYPE</b>          | Job Type, one of the following: <b>P</b> , <b>D</b> , <b>S</b> (program, database, script)  |
| <b>KEYSTROKE</b>         | Keystroke   |
| <b>LOG</b>               | Log Job Execution, one of the following: <b>Y</b> , <b>N</b> (yes, no)  |
| <b>MESSAGE</b>           | E-mail Message Text (e-mail watch job)  |
| <b>MODIFY_TERMINAL</b>   | Network name of the computer from which the job was last modified. This is a read-only property. It cannot be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command  |
| <b>MODIFY_TIME</b>       | Date and time when the job was modified. This is a read-only property. It cannot be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command  |
| <b>MODIFY_USER</b>       | Name of the user who last modified the job. This is a read-only property. It cannot be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command   |
| <b>MONDAY</b>            | Execute Job On Mondays, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>MONTHLY_TYPE</b>      | Monthly Schedule Type, one of the following: <b>T</b> , <b>D</b> (by day number, by day name)   |
| <b>MSG_ACCOUNT</b>       | E-mail Account for Notification Action of E-mail Type E-mail (user ID, profile, or e-mail address).<br>The actual value may differ for different e-mail interfaces. For the MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to Lotus Notes. For SMTP you should use your e-mail address. |
| <b>MSG_DATABASE</b>      | Execute Notification Action of Database Type, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>MSG_E-MAIL</b>        | Execute Notification Action of E-mail Type, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>MSG_ERROR</b>         | Execute Notification Action on Job Execution Error, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>MSG_FILE</b>          | Execute Notification Action of Semaphore File Type, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>MSG_FILE_NAME</b>     | File name(s) for Notification Action of Semaphore File Type   |
| <b>MSG_FINISH</b>        | Execute Notification Action on Job Finish, one of the following: <b>Y</b> , <b>N</b> (yes, no)  |
| <b>MSG_NOTFOUND</b>      | Execute Notification Action on Job Executable Not Found Error, one of the following: <b>Y</b> , <b>N</b> (yes, no)  |
| <b>MSG_PASSWORD</b>      | E-mail Password for Notification Action of E-mail Type  |
| <b>MSG_PROFILE</b>       | Database Profile for Notification Action of Database Type   |
| <b>MSG_RECIPIENT</b>     | E-mail Recipient for Notification Action of E-mail Type   |
| <b>MSG_SQL</b>           | SQL Command(s) for Notification Action of Database Type   |
| <b>MSG_START</b>         | Execute Notification Action on Job Start, one of the following: <b>Y</b> , <b>N</b> (yes, no)   |
| <b>NAME</b>              | Job Name  |
| <b>NUMBER_OF_RETRIES</b> | Maximum Number of Retries Before Job Gets Marked as Failed (number)   |
| <b>PASSWORD</b>          | E-mail Password (e-mail watch job)  |
| <b>POLLING_INTERVAL</b>  | Polling Interval (minutes)  |
| <b>PROFILE</b>           | Database Profile  |
| <b>PROTECTION</b>        | Job Protection State, one of the following: <b>F</b> , <b>E</b> , <b>R</b> , an empty string ( <b>F</b> – full protection; <b>E</b> – execute only; <b>R</b> – read only; an empty string indicates that a job is not protected). This is a read-only property. It cannot be changed using <b>SET</b> command. It can be retrieved using <b>GET</b> command.  |

|                        |  |
|------------------------|--|
| <b>REBOOT</b>          | Reboot Computer After Job Finished, one of the following: <b>Y, N</b> (yes, no)  |
| <b>RETRY_INTERVAL</b>  | Retry Interval (seconds)   |
| <b>RETRY_ON_ERROR</b>  | Retry Job After an Error Occurred, one of the following: <b>Y, N</b> (yes, no)   |
| <b>SATURDAY</b>        | Execute Job On Saturdays, one of the following: <b>Y, N</b> (yes, no)  |
| <b>SAVE_ATTACHMENT</b> | Save E-mail Attachments (e-mail watch job), one of the following: <b>Y, N</b> (yes, no)  |
| <b>SCHEDULE_TYPE</b>   | Schedule Type, one of the following: <b>O, D, T, M, F, P, A, E, I, L, S</b> (Time trigger: <b>O</b> – run once, <b>D</b> – repeat daily, <b>T</b> – repeat at specified time interval, <b>M</b> – repeat monthly; File trigger: <b>F</b> – check semaphore files; Process trigger: <b>P</b> – check process presence, <b>A</b> – check process absence; E-mail trigger: <b>E</b> - check e-mail message; User trigger: <b>I</b> – wake up on "user idle" event, <b>L</b> - wake up on log-off event, <b>S</b> – wake up on shutdown event) |
| <b>SCRIPT</b>          | JAL Script   |
| <b>SEND_KEYSTROKE</b>  | Send Keystroke, one of the following: <b>Y, N</b> (yes, no)  |
| <b>SKIP</b>            | Skip Late Job, one of the following: <b>Y, N</b> (yes, no)   |
| <b>SKIP_HOLIDAY</b>    | Skip Job on Holiday, one of the following: <b>Y, N</b> (yes, no)   |
| <b>SLIDE_HOLIDAY</b>   | Slide Job Execution Time on the next non-holiday if it falls on holiday, one of the following: <b>Y, N</b> (yes, no)   |
| <b>SQL</b>             | SQL Command(s)   |
| <b>START_DATE</b>      | First Start Date   |
| <b>START_IN</b>        | Program Start-up Directory   |
| <b>START_TIME</b>      | First Start Time   |
| <b>SUBJECT</b>         | E-mail Message Subject for (e-mail watch job)  |
| <b>SUNDAY</b>          | Execute Job On Sundays, one of the following: <b>Y, N</b> (yes, no)  |
| <b>THURSDAY</b>        | Execute Job On Thursdays, one of the following: <b>Y, N</b> (yes, no)  |
| <b>TIMEOUT</b>         | Timeout (seconds)  |
| <b>TUESDAY</b>         | Execute Job On Tuesdays, one of the following: <b>Y, N</b> (yes, no)   |
| <b>WEDNESDAY</b>       | Execute Job On Tuesdays, one of the following: <b>Y, N</b> (yes, no)   |
| <b>WINDOW</b>          | Window Style, one of the following: <b>N, M, I, H</b> (normal, maximized, iconic, hidden)  |

For information on properties description and usage see Job Properties topic.

**See also:**

- JDL Interface Examples
- JDL Overview
- Using JDL Files
- Using DDE

## Examples of Using Job Definition Language Commands

### 1. Command Files

The following is an example of using JDL file from command line interface includes. This example includes two files: EXAMPLE.JDL and JDL.BAT. Note that the JDL.BAT file is universal. You can use it in your projects as it is.

#### JDL.BAT

```
@echo off
echo the 24x7 Scheduler
echo Processing JDL command file %1%
24x7 /SCRIPT %1%
type jdl.log
```

### EXAMPLE.JDL

```
; Change name for the job #1
SET 1 NAME=JDL command file test 1
; Change start-in directory for the job #2
SET 2 STARTIN=c:\windows
; Add the new job to the schedule
ADD
    NAME=New job submitted via JDL command file
    COMMAND=notepad
    ASYNC=N
    TIMEOUT=1
    DESCRIPTION=This jobs was created using JDL command file
; Save all changes
SAV
```

To try this example:

- 1 Create both required files.
- 2 Make sure the 24x7 Scheduler is running.
- 3 From the DOS prompt type [JDL EXAMPLE.JDL](#).

## 2. DDE interface

This example shows how to use Visual Basic to execute JDL commands using DDE protocol.

```
Attribute VB_Name = "24x7"
Option Compare Database
Option Explicit

Sub Macro_24x7( )
'
' Establish DDE link to the 24x7 Scheduler and send some JDL commands
'
    Dim ChannelNumber As Long, NewJob As Long, JobName As String

    ' initiate DDE conversation
    ChannelNumber = Application.DDEInitiate("24x7 Scheduler" , "JDL" )

    ' get job #1 name
    JobName = DDEGetValue(ChannelNumber, "1"+ Chr(9) + "NAME" )

    ' disable job #1
    Call DDEDisable(ChannelNumber, "1")
    ' or alternatively perform the same operation using job name
    ' instead of using job number. This works for all commands
    ' Call DDEDisable(ChannelNumber, JobName)

    ' change job #1 name
    Call DDESetValue(ChannelNumber, "1"+ Chr(9) + "Name" , "New Name for job #1")
End Sub
```

```
' delete job #1
Call DDEDelete(ChannelNumber, "1")

' add new job
NewJob = DDEAdd(ChannelNumber)
If NewJob > 0 Then ' set new job properties
    Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Name" , "New Name"
)
    Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Command" ,
"c:\feed\mfeed.ex /S" )
    Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Start_Time" ,
"19:30")
    Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Async" , "N" )
    Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Timeout" , "30")
End If

' save all changes
'Call DDESave(ChannelNumber)

' terminate DDE conversation
Application.DDETerminate ChannelNumber
End Sub

Function DDEGetValue(ChannelNumber As Long, Location As String) As String
    DDEGetValue = Application.DDERequest(ChannelNumber, Location)
End Function

Sub DDESetValue(ChannelNumber As Long, Location As String, Data As String)
    Application.DDEPoke ChannelNumber, Location, Data
End Sub

Sub DDEDisable(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "DIS "+ Location
End Sub

Sub DDEEnable(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "ENA "+ Location
End Sub

Sub DDEDelete(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "DEL "+ Location
End Sub

Sub DDESave(ChannelNumber As Long)
    Application.DDEExecute ChannelNumber, "SAV"
End Sub

Function DDEAdd(ChannelNumber As Long) As Long
    Application.DDEExecute ChannelNumber, "ADD"
    DDEAdd = DDEGetValue(ChannelNumber, "NEW_ID" )
End Function
```

To try this example:

- 1 Copy the code to the application supporting Visual Basic for Applications programming language.
- 2 Make sure the 24x7 Scheduler is running.
- 3 Run the [Macro\\_24x7](#) macro.

**Important Note:**

*The above example will not work in Microsoft Excel due to Excel specifics*

## Chapter 18: System Options

The 24x7 Scheduler can be reconfigured to some extent. Select the **Tools/Options** menu to open the Options dialog. You can customize the following features:

### General Options

- **When the "start in" directory is not specified, use the program directory instead** – This instructs the 24x7 Scheduler to start the scheduled programs in the program working directories when the "start in" job property is empty. The working directory is the directory in which the executable program finds the associated files needed to run the program. Most programs do not need an entry in this field. Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. Check this box to allow the 24x7 Scheduler to change to the working directory when running a program or a document.
- **On startup, automatically adjust job start date and time** – If the 24x7 Scheduler is restarted (either manually, or because the system is restarted), it will adjust the scheduled time for the jobs that run repeatedly at the specified time interval. The adjustment is performed by adding the correct number of specified intervals so that the new start date and time exceed the 24x7 Scheduler restart time. Such jobs will then run normally.
- The **maximum error message display time** sets the maximum display time for an error message box before the message disappears. To disable appearance of error messages, set this parameter to zero.
- **Ask confirmation on exit** – Whether to prompt for confirmation on exiting 24x7 Scheduler. This is highly recommended since the prompt message may also include information on any active jobs and connections at the time of the exit. Exiting while a job is running or an active database operation is in progress, can cause unpredictable results. You should disable this prompt only in very special circumstances such as when a user input is not possible or undesirable. Note that rebooting the computer using either the 24x7 Scheduler's "Reboot after job" feature or the Reboot statement does not produce the confirmation prompt.
- **Automatically save changes on exit** – This allows saving any pending changes in the job database without being asked to confirm them.
- **Generate status report (HTML)** – This allows updating of the Status Report after each job run. The 24x7 Scheduler is capable of duplicating job execution log entries in an elegant HTML report format and automatically updating this report each time a new entry is added to the log file. This report can reside on your company Web server, allowing you to view the Status Report using any Web browser that supports frames. You will therefore be able to monitor job execution over the Internet.
- **Status Report directory** – This specifies the destination directory of the Status Report. Alternatively, you can click the **Browse** button to select the desired directory. If you leave the field blank, the 24x7 Scheduler will save reports in the installation directory. If your company Web server can be accessed via Intranet, you can specify the destination directory on the Web server to have the 24x7 Scheduler automatically upload the Status Report.

### Fail-Save Mode and Distributed Service Properties

You can use this set of properties to configure the 24x7 Scheduler's distributed features for the Master and Standby scheduler, as well as the Remote Agents. For more information on **Fail-over mode** and connection parameters see About Fail-Save Mode topic.

- **Multi-instance synchronization enabled** – Check this option to enable Master/Standby scheduler synchronization.
- **Synchronization interval** – This specifies the interval (in seconds) at which you want the Standby scheduler to connect to the Master scheduler and synchronize their job databases.

## Editor Options

- **Font** - Clicking on the **Font** button opens the Select Font dialog box. Use this option to customize font characteristics for the JAL and SQL Editors.
- **Automatically save a script recovery file** – This allows periodic saving of the backup copy of the script opened in the 24x7 Script Editor. You may want to use this option while you work on a script and you have 24x7 Scheduler running other jobs in background. If your computer hangs (stops responding) or you lose power unexpectedly, you can use the **AutoSave.jdl** file to recover your script the next time you start 24x7 Editor.
- **Autosave interval** – This specifies the interval (in minutes) at which you want the Editor to update the recovery file.
- **Search for special ASCII characters** – This allows 24x7 Scheduler to search for special ASCII characters escape symbols and substitute them with the appropriate ASCII characters.
- **Escape character** – This designates the character to be used as a prefix for special ASCII characters. Backslash (\) is the default escape character.

**Note:**

- **IMPORTANT:** AutoSave does not replace the Save command. You must still save your job definition and script changes when you finish working on them.

## Log and Debug Options

- **Load log on startup** – This instructs the 24x7 Scheduler to load the job execution log on startup. Clearing this check box will disable log loading and reduce 24x7 Scheduler loading time. However, you will not be able to see old log entries in the Job Explorer.
- **Maximum number of entries in the log file** – The depth of the job history in the log is limited by the **maximum number of entries in the log** parameter. The 24x7 Scheduler, for performance reasons, keeps the log loaded in the computer memory. Therefore, the amount of memory required depends on how many records you have in that log. Under normal circumstances, you should let the 24x7 Scheduler capture the job history for at least a week. This will allow you to view the historical status of your jobs. If however, you created a job that runs frequently, such as once each minute, you should not allow large logs and set the **maximum number of entries in the log** parameter to a reasonable value.
- **Clear log on startup** – This instructs the 24x7 Scheduler to empty job execution logs on startup. Clear this check box to disable clearing of the job history. Under normal circumstances, you should let the 24x7 Scheduler capture the job history for at least a week. However, clearing large logs will save some disk space.

In addition, you can set the following properties used to troubleshoot job execution and communications between 24x7 Scheduler components.

- **Trace enabled** – Enables tracing. This option is used to:
  - 1 Troubleshoot Standby/Master Scheduler connections. When tracing is enabled, 24x7 Scheduler traces all activity between the Standby and the Master components. This internal information can be helpful when debugging, including memory usage, an internal call trace, and the types and values for passed parameters.
  - 2 Troubleshoot connections to Remote Agents. When tracing is enabled, 24x7 Scheduler traces all activity between the main scheduler and the Remote Agent. This internal information can be helpful when debugging, including memory usage, an internal call trace, and the types and values for passed parameters.
  - 3 Debug JAL script execution. When tracing is enabled, 24x7 Scheduler traces all executed script statements including statement parameters and returned values. This internal information can be helpful when debugging JAL scripts. Some JAL statements also output additional information that be visible only in the trace.
  - 4 Debug external interfaces, both command line and DDE interfaces. When tracing is enabled, 24x7 Scheduler traces all received commands including command parameters and internal information on command processing. This information can be helpful when troubleshooting external interfaces.
- **Database trace enabled** – Enables database tracing. This option is used to record the internal commands performed by 24x7 Scheduler while communicating with a database. This internal information can be helpful when troubleshooting database operations.

- **Job execution statistics enabled** (Windows NT only) – Enables collection of Job Execution statistics. You can use collected execution statistics to better understand the behavior of scheduled programs, such as processors and memory usage, run-time duration and delays. This information may also help you when troubleshooting job anomalies.
- **Logging on to the system event log enabled** (Windows NT only) – Enables parallel logging of all job log entries in the Windows NT Application Log. For detail on the Windows NT Application Log, click **Start** button, then choose **Help** command from the Windows NT Start menu.
- **Job execution status display enabled** - Enables displaying of the status/progress message box while executing scheduled jobs. The status/progress message box allows you to watch job execution progress.

## Service Options for Windows NT

This set of options is available on Windows NT platform only

- **Run 24x7 as a Windows NT service** – Check this option to install the 24x7 schedule service. Uncheck this options to uninstall the 24x7 schedule service.
- **Service account name** - Name of Windows NT account for which the 24x7 schedule service will be installed. This must be either the system account specified as "LocalSystem" or a user account specified using ".\<Local User>"format (where <Local user> must be substituted with actual account name) or "<Domain>\<User>"format.
- **Service start options** – Enables/disable the 24x7 schedule service to start automatically when the specified user logs on. When local system account is selected and "Start automatically" options is chosen, the 24x7 Scheduler starts automatically whenever the computer is started and runs continuously in the background, regardless of whether a user is logged on.
- **Service mode** – Specifies whether the 24x7 schedule service functions as a standalone 24x7 Scheduler or a 24x7 Remote Agent.



### Notes:

- You can also modify parameters of an existing 24x7 schedule service by using the Services Control Panel Applet.
- You must have system administrator privileges in order to install, uninstall, or modify the 24x7 schedule service.

## Service Options for Windows 95/98

This set of options is available on Windows 95 and Windows 98 platforms only

- **Run 24x7 as a Windows 95/98 service** – Check this option to install the 24x7 schedule service. Uncheck this options to uninstall the 24x7 schedule service. When the service is installed, 24x7 Scheduler starts automatically whenever the computer is started.
- **Survive user log off** – Check this option if you do not want 24x7 to exit on "Close all programs and log on as a different user" Windows operation
- **Service start options** – Always "Start automatically".
- **Service mode** – Specifies whether the 24x7 schedule service functions as a standalone 24x7 Scheduler or a 24x7 Remote Agent.

## Chapter 19: 24x7 Job Automation Language

### Overview

You write job automation scripts using Job Automation Language (JAL), the 24x7 Scheduler language. JAL provides you with great flexibility in scheduling and executing different kinds of jobs. It allows you to build complex job dependencies; automate and control the execution of other programs with ability to simulate an operator by sending various keystrokes to the program being executed as well as perform advanced error checking and handling. JAL includes over 250 statements for a wide variety of operations such as file manipulations, program execution, process handling, database operations, and many other.

As with other parts of the 24x7 Scheduler, you can use macro-parameters in your JAL scripts. Macro-parameter values are substituted immediately before the JAL interpreter starts script execution.

The 24x7 Scheduler provides you with a built-in JAL editor that you can use to create new, as well as modify existing, JAL and SQL scripts.

#### See also:

Syntax

Statements by Category

### Syntax

The 24x7 Job Automation Language supports the following three programming language elements:

- 1 Variables
- 2 Statements
- 3 Comments

### Variables

A JAL variable is a named storage location that contains data that can be modified during program execution. Each variable has a unique name that identifies it within the JAL script. You must declare a variable before you can use it. Use the [Dim](#) statement to declare variables and their data types. When you declare a variable, you can accept the default initial value or specify an initial value.

The variable name should begin with an alphabetic character, followed by any combination of alphabetic characters, numbers, or the underscore character, as in this example: [ProcessID](#).

A variable can be declared as one of the following data types:

- **NUMBER** - A signed floating-point number with 15 digits of precision and a range from 2.2250738585072E-308 to 1.79769313486232E+308. When necessary, the 24x7 Scheduler automatically handles conversion between floating-point and integer numbers. Variables of NUMBER data type have [0](#) (zero) as their default initial value.
- **STRING** - Any string of ASCII characters with variable length (0 to 2,147,483,647). Variables of STRING data type have [""](#) (empty string) as their default initial value.
- **DATE** - The date, including the full year (1000 to 3000), the number of the month (01 to 12), and the day (01 to 31). Variables of DATE data type have [1900-01-01](#) (January 1, 1900) as their default initial value.
- **TIME** - The time in 24-hour format, including the hour (00 to 23), minute (00 to 59), and second (00 to 59), with a range from 00:00:00 to 23:59:59. Variables of TIME data type have [00:00:00](#) (midnight) as their default initial value.
- **DATETIME** - The date and time in a single data type. Variables of DATETIME data type have [1900-01-01 00:00:00](#) (January 1, 1900 midnight) as their default initial value.
- **BOOLEAN** - Contains TRUE or FALSE. Variables of BOOLEAN data type have [FALSE](#) as their default initial value.

A variable can have either local or global scope. A local variable is a temporary variable accessible only from within the script in which you define it. When the script is finished, the variable constant ceases to exist. Global variables are accessible from anywhere in any script of any job. To avoid ambiguity when referring to variables, you must use the **GLOBAL** prefix when declaring or referring to global variables. The 24x7 Scheduler allocates memory and initializes a global variable whenever it executes a JAL script, in which it first time finds the declaration of that variable. 24x7 Scheduler ignores the declaration on subsequent runs. You can use the **Set** statement to change the values of global variables.

Examples:

This statement declares a local variable: **Dim( counter, number )**

This statement declares a global variable: **Dim( global.counter, number )**

This statement sets the value of a local variable: **Set( count, 5 )**

This statement sets the value of a global variable: **Set( global.count, 5 )**

## Statements

JAL statements have two main components: the statement name and the statement parameters. The statement and variable names are case insensitive. The statement name must be spelled exactly as it is shown in the syntax and the parameters must be used in the order they are given in the syntax. Parameters provide information for the statement. For example, the **FileReadLine** statement, which reads a line from a text file, has parameters for the name of the file, the line number, and the variable name into which the line will be stored.

All JAL statements use the following syntax:

**StatementName(parameter1, parameter2, ...)**

The opening parenthesis, closing parenthesis, and commas are optional. You can use spaces and tabs instead. Statement names are not case-sensitive. The 24x7 Scheduler automatically removes all unnecessary white space before executing JAL script. This enables you to use spaces and tabs in the script to improve readability.

Parameters can be variable names, text strings or numbers, and are separated by commas, spaces, and/or tabs. Text string parameters must be enclosed in double quotes. The number of white space characters is preserved when they are part of a string literal (parameter).

Some statements have return value. The return value is always passed in the last statement parameter. When coding statements you should declare a variable of the appropriate type in order to store this return value.

Some statements have no parameters at all. For example: **Exit**

You cannot nest more than one statement in a statement string.

@ sign is used as an escape character that allows you to include **macro-variables** as part of a statement parameter. If you need a literal @ sign (for example, in a string specification) you must double the @ sign.

You can define your own JAL statements. You can use the **Script Library** tool (Tools/Script Library menu) to create, modify, and delete your **user-defined JAL statements**. User-defined JAL statements may call other user-defined statements. You should be very careful when coding your JAL statements and avoid cyclical calls that may lead to infinite loops.

## Statement continuation

Although typically you would put one statement on each line, you will occasionally want to continue a statement on to more than one line. The statement continuation character is the ampersand (&).

Syntax:

*Start of statement &  
more statement &  
end of statement*

The ampersand must be the last nonwhite character on the line (or the script interpreter will consider it as a part of the statement).

## Special ASCII characters

You can include special ASCII characters in strings. For example, you may want to include a tab in a string to ensure proper spacing or a new line character to indicate end of line. The backslash character (\) introduces special characters. To specify the backslash character as string literal, precede it with another backslash. Instead of the backslash character (\) you can specify another character to be used as a prefix for special characters. See 24x7 Scheduler Tools/Options for details.

Syntax:

| <b>Symbol</b>   | <b>ASCII character</b> |
|-----------------|------------------------|
| <code>\n</code> | <i>New line</i>        |
| <code>\r</code> | <i>Carriage return</i> |
| <code>\t</code> | <i>Tab</i>             |
| <code>\"</code> | <i>Double quote</i>    |
| <code>\\</code> | <i>Backslash</i>       |

## Off-line scripts

You can develop off-line JAL scripts that are saved in text files as opposed to in-line scripts that are stored in the job database. You use the [@SCRIPT](#) tag in in-line scripts to reference off-line scripts. In run-time, 24x7 Scheduler inserts the contents of the off-line script file into the body of the main job script.

Syntax:

`@SCRIPT:<file name>`

For example: [@SCRIPT:c:\scripts\check\\_errors.jal](#)

## Comments

A comment is a line that starts with // (double forward slash). You cannot use comments on the line that already has some statement or statement. You can have as many comments as you want.



### Important Notes:

- JAL script lines cannot exceed 8K (8192 characters) in size.
- The total script size cannot exceed 32000 characters.
- Nested statements are not currently supported.

### See also:

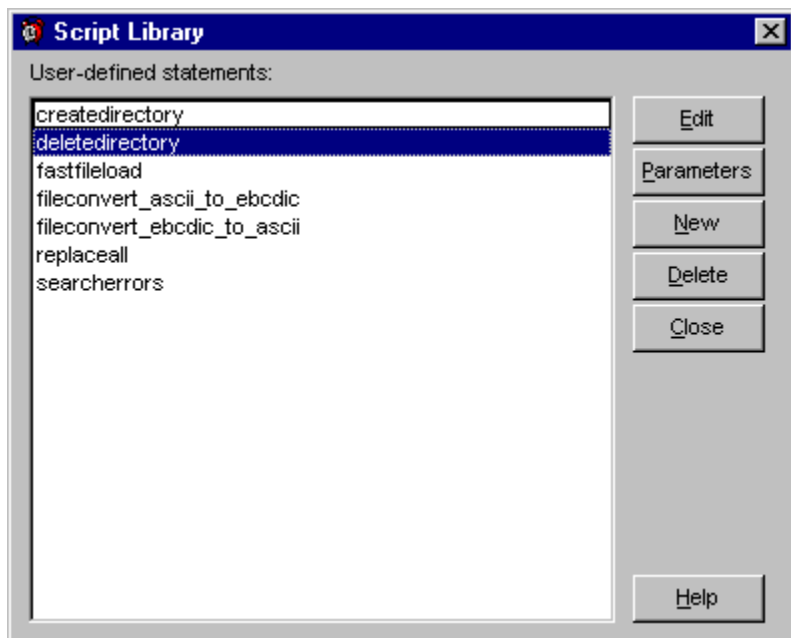
Statements by Category

Functions in Filter and Sort Expressions

## Script Library

You can access the Script Library from the **Tools/Script Library** menu or simply by pressing the F7 hot key.


The Script Library is the place where you can create and store your user-defined JAL statements. User-defined JAL statements have global scope and can be called from any script type job. You can call them from a script just as you would call any other built-in JAL statement. From a user-defined JAL statement you can also call other user-defined statements and jobs. Maintenance of multiple jobs can be made a lot easier by implementing common business functions as user-defined statements.



User-defined statements must follow the same syntax rules that apply to other JAL scripts. Just as script jobs, variables declared in any user-defined JAL statement have local scope unless they are declared as **GLOBAL**. User-defined statements, just like built-in JAL statements, may have parameters. Parameters are defined at the time when you declare new user-defined statement in the Script Library. Every parameter must have a name. Parameters are always passed by value and inside the user-defined statement they are treated as local variables. In the statement code, you can refer to them by their names. However, their names are not used when calling the statement from some other script. The position of the parameter in the user-defined statement is important. The order in which you specify parameter is the order you will use when calling the statement.

If you declare your user-defined statement with the return value, you must use the **Return** statement to return some value to the calling script. To call such user-defined statement, specify appropriate variable for the return value as the last parameter in the statement call line.

For example: `MyStatement first_param_as_var, "abc", 123, "third param as string", ret_value_var.`

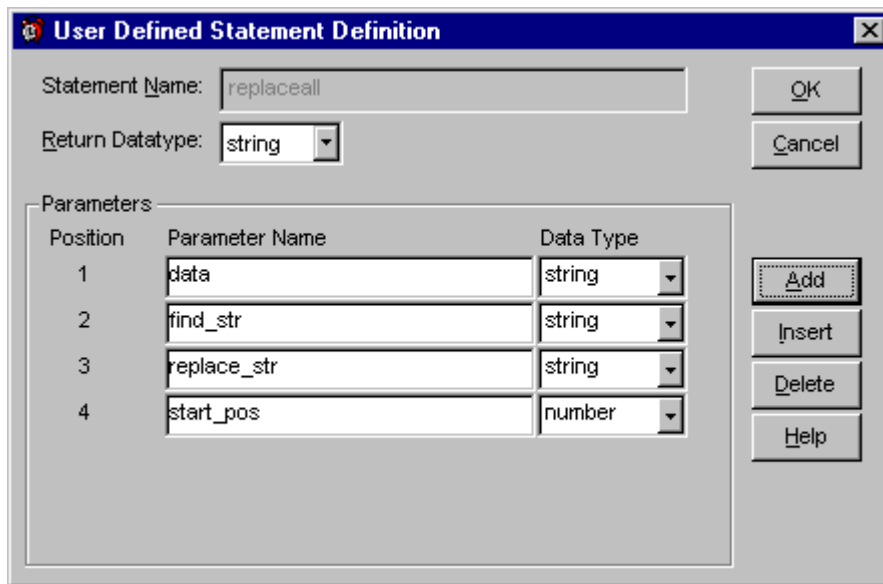
 **Caution:** User-defined statements can be recursive; that is, they can call themselves to perform a given task. However, recursion can lead to stack overflow. Be very careful when implementing recursive calls.

JAL is not case sensitive, that is, when calling your user-defined statements you can specify their names in lower, upper, or mixed case.

#### To create a new user-defined statement:

- 1 Open **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
- 2 Click the **New** button. The Statement Definition dialog box will appear.
- 3 Specify the new statement name. The name must follow standard naming convention and must not exceed 50 characters in length. Generally, you should not use names that are the same as the names of the built-in JAL statements. Otherwise, you would shadow the same keywords in the language.
- 4 Specify the return value type or leave the return value (**None**) if the new statement has no return value.
- 5 Specify the statement parameter names and their data types. Parameter names must follow standard naming convention and must not exceed 50 characters in length. You can specify as many parameters as needed. To add more parameters, use the **Add** button.
- 6 When you are finished defining the parameters, click the **OK** button. The Script Editor window will appear.

- 7 Code your user-defined statement then click the **Exit** button to close the Script Editor and return to the Script Library.



**To modify an existing user-defined statement:**

- 1 Open the **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
- 2 Find and select the desired statement.
- 3 To modify statement parameters, click the **Parameter** button. The Statement Definition dialog box will appear.
- 4 To modify the statement code, click the **Edit** button. The Script Editor window will appear.
- 5 Code your user-defined statement then click the **Exit** button to close the Script Editor and return to the Script Library.

**To delete an existing user-defined statement:**

- 1 Open the **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
- 2 Find and select the desired statement.
- 3 Click the **Delete** button.

## Control-of-Flow Statements

24x7 Job Automation Language includes the following statements for controlling script execution logic, declaring variables, and assigning values to variables:

---

### Break

**Description:** In a [LoopWhile](#), [LoopUntil](#), or a [ForNext](#) control structure, passes control out of the current loop. [Break](#) takes no arguments.

**Syntax:** [Break](#)

**Usage:** A [Break](#) statement in a loop control structure causes control to pass to the statement following the end of loop label. In a nested loop, a Break statement passes control out of the current loop structure. For information on how to jump to the end of the loop and continue looping, see [Continue](#) statement.

---

### Continue

**Description:** In a [LoopWhile](#), [LoopUntil](#), or a [ForNext](#) control structure, skips statements in the current loop. [Continue](#) takes no arguments.

**Syntax:** [Continue](#)

**Usage:** When 24x7 Scheduler encounters a [Continue](#) statement in a loop structure, control passes to the loop's end label. The statements between the [Continue](#) statement and the loop's end label are skipped in the current iteration of the loop. In a nested loop, a [Continue](#) statement bypasses statements in the current loop structure. For information on how to break out of the loop, see [Break](#) statement.

---

### Dim

**Description:** Declares variables and allocates storage space.

**Syntax:** [Dim](#) [variable](#), [datatype](#) [, [initial\\_value](#)]

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">variable</a> | The name of the <a href="#">variable</a> you want to declare                            |
| <a href="#">datatype</a> | The data type of the <a href="#">variable</a> . The following data types are supported: |

|                            |  |
|----------------------------|--|
|                            | <ul style="list-style-type: none"><li>• String</li><li>• Number</li><li>• Boolean</li><li>• Date</li><li>• Time</li><li>• DateTime</li></ul>   |
| <code>initial_value</code> | The initial value of the <code>variable</code> . The data type of the <code>initial_value</code> must match the declared data type of the <code>variable</code> . This argument is optional. |

**Usage Notes:** To declare a global variable, use *dot notation* with the `GLOBAL` prefix in front of the variable name.

---

## Exit

**Description:** Immediately exits the script.

**Syntax:** `Exit`

**Usage:** This statement has no arguments.

---

## RaiseError

**Description:** Causes an error event at the job level and writes error message to the 24x7 event log.

**Syntax:** `RaiseError error`

| Argument           | Description   |
|--------------------|---|
| <code>error</code> | A string whose value is the error message that you want to write to the 24x7 event log. |

**Usage:** `RaiseError` statement halts job execution and writes `error` message to the 24x7 event log. If parallel logging to the Windows NT event log enabled, `RaiseError` also writes `error` message to the Windows NT application event log.

**See also:**

LogAddMessageEx  
Exit

## ForNext

**Description:** A control structure that is a numerical iteration, used to execute one or more statements a specified number of times.

**Syntax:** `ForNext variable, start, end, step, label`  
`statementblock`  
`label:`

| Argument | Description  |
|----------|--|
| Variable | The name of the iteration counter variable having <code>number</code> data type.   |
| Start    | Starting value of <code>variable</code> . It can be a constant or name of another variable.  |
| End      | Ending value of <code>variable</code> . It can be a constant or name of another variable.  |
| Step     | The increment value. It can be a constant or name of another variable.   |
| Label    | An end loop <code>label</code> is an identifier followed by a colon (such as OK:). This <code>label</code> followed by a colon must be placed on a separate line. Do not use the colon with a label in the <code>LoopWhile</code> statement. |

**Usage:**

- Using the `start` and `end` parameters: for a positive increment (`step`), `end` must be greater than `start`; for a negative increment, `end` must be less than `start`, otherwise it will lead to an infinite loops. When increment is positive and `start` is greater than `end`, `statementblock` does not execute. When increment is negative and `start` is less than `end`, `statementblock` does not execute.
- Use `Break`, `GoTo` or `If` statement to jump out of the loop structure. Note that `Exit` or `Reboot` statements will also interrupt the loop.
- Use `Break` statement to interrupt the loop and continue with the first statement that follows the loop end `label`.
- Use `GoTo label` inside loop structure (where the label is the same as the label referenced in `ForNext`) to jump to the end of the loop and continue looping. You can also use `Continue` statement for this purpose.
- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.
- The `label` name must satisfy Universal Naming Convention (UNC).



**Note:**

When `start` and `end` are variables, they are reevaluated on each pass through the loop. So if the variable's value changes, it will affect the number of loops. Consider this example—the body of the loop changes the number of rows in the database buffer, which changes the result of the `DatabaseRowCount` statement:

```
...
DatabaseRowCount( row_count )
ForNext n, 1, row_count, 1, END_LOOP
    DatabaseDelete( 1 )
    Subtract( row_count, 1, row_count )
END_LOOP:
...
```

**See also:**

LoopUntil  
LoopWhile  
Break

Continue

---

## GoTo

**Description:** Transfers control from this statement in a script to another statement or statement that is preceded by a label.

**Syntax:** `GoTo label`

| Argument | Description   |
|----------|---|
| Label    | The <code>label</code> associated with the statement or statement to which you want to transfer control. A <code>label</code> is an identifier followed by a colon (such as OK:). This <code>label</code> followed by a colon must be placed on a separate line. Do not use the colon with a label in the <code>GoTo</code> statement. A <code>label</code> |

**Usage:** The `label` name must satisfy Universal Naming Convention (UNC).

---

## If

**Description:** Evaluates a conditional Boolean variable and depending on its value transfers control to the statement following the specified labels.

**Syntax:**

```
If ( boolean, truelabel, falselabel )
    statementblock
truelabel:
    statementblock
falselabel:
    statementblock
```

| Argument       | Description   |
|----------------|---|
| Boolean        | The <code>boolean</code> variable that evaluates to TRUE or FALSE. If the <code>boolean</code> variable evaluates to TRUE then control will be transferred to the statement going after <code>truelabel</code> , otherwise control will be transferred to the statement going after <code>falselabel</code> . |
| statementblock | The block of JAL statements you want 24x7 Scheduler to execute.   |
| Truelabel      | A <code>truelabel</code> is an identifier followed by a colon (such as OK:). This <code>truelabel</code> followed by a colon must be placed on a separate line  |
| Falselabel     | A <code>falselabel</code> is an identifier followed by a colon (such as OK:). This <code>falselabel</code> followed by a colon must be placed on a separate line.   |

**Usage:**

You can use **If** statements to branch program logic that depends on some conditions. The **truelabel** and **falselabel** names must satisfy Universal Naming Convention (UNC).

**Notes:**

- Instead of the **boolean** value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.
- After executing block of JAL statements following **truelabel** 24x7 Scheduler executes section following **falselabel** unless you code some other Control-Of-Flow statement to transfer control beyond **falselabel** section.

**See also:**

IfThen

---

## IfThen

**Description:** Evaluates a conditional Boolean variable and depending on its value transfers control to the next statement in script or the statement following the specified label.

**Syntax:**

```
IfThen ( boolean, label)
    statementblock
label:
```

| Argument              | Description   |
|-----------------------|---|
| <b>boolean</b>        | The <b>boolean</b> variable that evaluates to TRUE or FALSE. If the <b>boolean</b> variable evaluates to FALSE then control will be transferred to the next statement going after <b>IfThen</b> , otherwise control will be transferred to the statement going after the <b>label</b> . |
| <b>statementblock</b> | The block of JAL statements you want 24x7 Scheduler to execute if the <b>boolean</b> value is <b>False</b> .  |
| <b>label</b>          | A <b>label</b> is an identifier followed by a colon (such as OK:). This <b>label</b> followed by a colon must be placed on a separate line.   |

**Usage:**

You can use **If** statements to branch program logic that depends on some conditions. The **label** name must satisfy Universal Naming Convention (UNC).

**Note:**

Instead of the **boolean** value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

If

## ChooseCase

**Description:** Evaluates a conditional variable and depending on its value transfers control to the appropriate "Case" section.

**Syntax:**

```
ChooseCase testvariable, label
Case valuelist1
    statementblock
[Case valuelist2]
    statementblock
[Case ...]
    statementblock
[CaseElse]
    statementblock
label:
```

| Argument       | Description  |
|----------------|--|
| testvariable   | The variable whose value is evaluated in the following <a href="#">Case</a> sections.  |
| Case           | The keyword beginning the new <a href="#">Case</a> section. You can define as many <a href="#">Case</a> sections as needed.  |
| valuelist      | One of the following: <ul style="list-style-type: none"><li>• A single value</li><li>• A list of values separated by commas (such as 2, 4, 6, 8)</li><li>• A single variable</li><li>• A list of variables separated by commas (such as VAR1, VAR2, VAR5)</li><li>• Any combination of the above with an implied OR between items (such as 1, 3, VAR2, 7, 9, VAR5)</li></ul> Data types of values and variables in the list must match data type of the conditional <a href="#">testvariable</a> |
| statementblock | The block of JAL statements you want 24x7 Scheduler to execute if the value of the <a href="#">testvariable</a> matches the value in <a href="#">valuelist</a> of the corresponding <a href="#">Case</a> section.  |
| CaseElse       | The keyword beginning the <a href="#">CaseElse</a> section to which control is transferred when the value of the <a href="#">testvariable</a> does not match any of the <a href="#">valuelist</a>  |
| label          | A <a href="#">label</a> is an identifier followed by a colon (such as OK:). This <a href="#">label</a> followed by a colon must be placed on a separate line.  |

**Usage:**

You can use [ChooseCase](#) statements to branch program logic that depends on some conditions. The [label](#) name must satisfy Universal Naming Convention (UNC).

At least one [Case](#) section is required. You must end a [ChooseCase](#) control structure with the [label](#).

If the value of the [testvariable](#) at the beginning of the [ChooseCase](#) statement matches a value in [valuelist](#) for a [Case](#) section, the statements immediately following that [Case](#) section are executed. Control then passes to the first

statement after the [label](#).

If multiple [Case](#) section exist, then [testvariable](#) is compared to each [valuelist](#) until a match is found or the [CaseElse](#) section or [label](#) is encountered.

[CaseElse](#) section is optional. If there is a [CaseElse](#) section and the [testvariable](#) value does not match any of the [valuelist](#), statemens in the [CaseElse](#) Section are executed. If no [CaseElse](#) section exists and a match is not found, the first statement after the [label](#) is executed.

**Note:**

Data type of the [testvariable](#) can be any of JAL standard data types. Data types of values and variables in the [valuelist](#) must much data type of the [testvariable](#)

**See also:**

If  
IfThen

---

## LoopWhile

**Description:** A control structure that is a general-purpose iteration statement used to execute a block of statements while a conditional Boolean variable evaluates to TRUE.

**Syntax:**

```
LoopWhile boolean, label
    statementblock
label:
```

| Argument                       | Description   |
|--------------------------------|---|
| <a href="#">Boolean</a>        | The <a href="#">boolean</a> variable that evaluates to TRUE or FALSE. If the <a href="#">boolean</a> variable evaluates to TRUE then control will be transferred to the statement going after <a href="#">LoopWhile</a> , otherwise control will be transferred to the statement or statement going after <a href="#">label</a> . |
| <a href="#">Statementblock</a> | The block of statements you want to repeat.   |
| <a href="#">Label</a>          | An end loop <a href="#">label</a> is an identifier followed by a colon (such as OK:). This <a href="#">label</a> followed by a colon must be placed on a separate line.   |

**Usage:**

- You can use [LoopWhile](#) statements to branch program logic that depends on some conditions.
- Avoid infinite loops. Make sure to change the conditional Boolean variable so it evaluates to FALSE or use [Break](#), [GoTo](#) of [If](#) statement to jump out of the loop structure. Note that [Exit](#) or [Reboot](#) statements will also interrupt the loop.
- Use [Break](#) statement to interrupt an endless loop and continue with the first statement that follows the loop end label.
- Use [GoTo label](#) inside loop structure (where the label is the same as the label referenced in [LoopWhile](#)) to jump to the end of the loop and continue looping. You can also use [Continue](#) statement for this purpose.
- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.
- The [label](#) name must satisfy Universal Naming Convention (UNC).

**Note:**

Instead of the [boolean](#) value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

LoopUntil  
ForNext  
Break  
Continue

---

## LoopUntil

**Description:** A control structure that is a general-purpose iteration statement used to execute a block of statements while a conditional Boolean variable does not evaluate to TRUE.

**Syntax:**

[LoopUntil](#) [boolean](#), [label](#)  
    [statementblock](#)  
[label](#):

| Argument                       | Description  |
|--------------------------------|--|
| <a href="#">Boolean</a>        | The <a href="#">boolean</a> variable that evaluates to TRUE or FALSE. If the <a href="#">boolean</a> variable evaluates to FALSE then control will be transferred to the statement going after <a href="#">LoopUntil</a> , otherwise control will be transferred to the statement or statement going after <a href="#">label</a> . |
| <a href="#">Statementblock</a> | The block of statements you want to repeat.  |
| <a href="#">Label</a>          | An end loop <a href="#">label</a> is an identifier followed by a colon (such as OK:). This <a href="#">label</a> followed by a colon must be placed on a separate line.  |

**Usage:**

- You can use [LoopUntil](#) statements to branch program logic that depends on some conditions.
- Avoid infinite loops. Make sure to change the conditional Boolean variable so it evaluates to TRUE or use [Break](#), [GoTo](#) or [If](#) statement to jump out of the loop structure. Note that [Exit](#) or [Reboot](#) statements will also interrupt the loop.
- Use [Break](#) statement to interrupt an endless loop and continue with the first statement that follows the loop end label.
- Use [GoTo label](#) inside loop structure (where the label is the same as the label referenced in [LoopUntil](#)) to jump to the end of the loop and continue looping. You can also use [Continue](#) statement for this purpose.
- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.

**Note:**

Instead of the [boolean](#) value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

LoopWhile  
ForNext

Break  
Continue

---

## Set

**Description:** Assigns new value to a variable.

**Syntax:** `Set variable, value`

| Argument              | Description  |
|-----------------------|--|
| <code>variable</code> | The name of <code>variable</code> you want to change   |
| <code>value</code>    | The new <code>value</code> that will be assigned to the <code>variable</code> . The <code>value</code> data type must be the same as <code>variable</code> data type |

**Usage:** The following data types are supported: String, Number, Boolean, Date, Time, DateTime

---

## Bitwise statements

---

### BitwiseAnd

**Description:** Performs a bitwise AND operation on each bit of two passed values.

**Syntax:** `BitwiseAnd value1, value2, return`

| Argument            | Description  |
|---------------------|--|
| <code>value1</code> | A first number whose value is used in the bitwise AND operation  |
| <code>value2</code> | A second number whose value is used in the bitwise AND operation |
| <code>return</code> | A numeric variable that receives the returned value              |

**Return value:** Number. Returns the result of the AND operation on each of `value1`'s and `value2`'s bits.

**Usage:** In a bitwise AND operation:

1 AND 0 evaluates to 0  
0 AND 0 evaluates to 0  
0 AND 1 evaluates to 0  
1 AND 1 evaluates to 1

**See also:**

Bitwise statements

---

## BitwiseClearBit

**Description:** Clears the specified bit in a number.

**Syntax:** [BitwiseClearBit value, bit, return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">value</a>  | A number whose value is used in the bitwise operation         |
| <a href="#">bit</a>    | A number whose value is the bit number that you want to clear |
| <a href="#">return</a> | A numeric variable that receives the returned value           |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is off ([0](#)), the returned value equals the original [value](#).

**See also:**

Bitwise statements

---

## BitwiseFlipBit

**Description:** Reverses the specified bit in a number.

**Syntax:** [BitwiseFlipBit value, bit, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">value</a>  | A number whose value is used in the bitwise operation        |
| <a href="#">bit</a>    | A number whose value is the bit number that you want to flip |
| <a href="#">return</a> | A numeric variable that receives the returned value          |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is off ([0](#)), [BitwiseFlipBit](#) sets the bit on ([1](#)), otherwise it clears the bit.

**See also:**

Bitwise statements

## BitwiseGetBit

**Description:** Reports whether the specified bit is on (1) or off (0).

**Syntax:** BitwiseGetBit value, bit, return

| Argument | Description  |
|----------|--|
| value    | A number whose value is used in the bitwise operation        |
| bit      | A number whose value is the bit number that you want to test |
| return   | A boolean variable that receives the returned value          |

**Return value:** Boolean. Returns TRUE if the bit is on (1) and FALSE if it is off (0).

**See also:**

Bitwise statements

---

## BitwiseOr

**Description:** Performs a bitwise OR operation on each bit of two passed values.

**Syntax:** BitwiseOr value1, value2, return

| Argument | Description   |
|----------|---|
| value1   | A first number whose value is used in the bitwise OR operation  |
| value2   | A second number whose value is used in the bitwise OR operation |
| return   | A numeric variable that receives the returned value             |

**Return value:** Number. Returns the result of the OR operation on each of value1's and value2's bits.

**Usage:** In a bitwise OR operation:

1 OR 0 evaluates to 1  
0 OR 0 evaluates to 0  
0 OR 1 evaluates to 1  
1 OR 1 evaluates to 1

**See also:**

Bitwise statements

---

---

## BitwiseNot

**Description:** Performs a bitwise NOT operation on each bit of a passed value.

**Syntax:** `BitwiseNot value, return`

| Argument            | Description   |
|---------------------|---|
| <code>value</code>  | A number whose value is used in the bitwise NOT operation |
| <code>return</code> | A numeric variable that receives the returned value       |

**Return value:** Number. Returns the result of the bitwise NOT operation on each of `value`'s bits.

**Usage:** In a bitwise NOT operation:

`1` evaluates to `0`

`0` evaluates to `1`

**See also:**

Bitwise statements

---

## BitwiseSetBit

**Description:** Sets the specified bit on (`1`) in a number.

**Syntax:** `BitwiseSetBit value, bit, return`

| Argument            | Description   |
|---------------------|---|
| <code>value</code>  | A number whose value is used in the bitwise operation       |
| <code>bit</code>    | A number whose value is the bit number that you want to set |
| <code>return</code> | A numeric variable that receives the returned value         |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is on (`1`), the returned value equals the original `value`.

**See also:**

Bitwise statements

---

## BitwiseXor

**Description:** Performs a bitwise XOR operation on each bit of two passed values.

**Syntax:** `BitwiseXor value1, value2, return`

| Argument            | Description  |
|---------------------|--|
| <code>value1</code> | A first number whose value is used in the bitwise XOR operation  |
| <code>value2</code> | A second number whose value is used in the bitwise XOR operation |
| <code>return</code> | A numeric variable that receives the returned value              |

**Return value:** Number. Returns the result of the XOR operation on each of `value1`'s and `value2`'s bits.

**Usage:** In a bitwise XOR operation:

1 XOR 0 evaluates to 1  
0 XOR 0 evaluates to 0  
0 XOR 1 evaluates to 1  
1 XOR 1 evaluates to 0

**See also:**

Bitwise statements

---

## Clipboard statements

---

### ClipboardGet

**Description:** Obtains a copy of the text stored in the system clipboard.

**Syntax:** `ClipboardGet return`

| Argument            | Description  |
|---------------------|--|
| <code>return</code> | A string variable that receives the Clipboard data |

**Return value:** String. Returns a copy of the text data stored in the system clipboard if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

ClipboardSet

## ClipboardSet

**Description:** Replaces a contents of the system clipboard with the specified text.

**Syntax:** [ClipboardSet string](#)

| Argument               | Description                                  |
|------------------------|--|
| <a href="#">string</a> | The string you want to copy to the clipboard |

**Return value:** None.

**See also:**

ClipboardGet

---

## Database statements

---

### DatabaseConnect

**Description:** Connects to a database using the specified profile.

**Syntax:** [DatabaseConnect profile](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">profile</a> | A string whose value is the name of the profile defined in the 24x7 Preferences |

**Return value:** None.

**Usage:** [DatabaseConnect](#) statement obtains from the profile all the required connection information for the database to which you want to connect. This statement must be executed before other database actions can be processed using the same profile. Only one database connection may be opened at a time. All other database statements executed after [DatabaseConnect](#) are sent to the database specified in the [profile](#).

**See also:**

DatabaseDisconnect

Database Profiles

## DatabaseConnectEx

**Description:** Connects to the database using the specified connection parameters.

**Syntax:** [DatabaseConnectEx dbms, server\\_name, database\\_name, user\\_id, password, auto\\_commit](#)

| Argument                                    | Description   |
|---|---|
| <a href="#">DBMS</a>                        | A string whose value is the type of the database management system that you want to connect to.<br><br>This name must match any supported DBMS name as they are specified in the Database Profiles dialog (see Tools/Database Profiles menu)  |
| <a href="#">server_name</a><br>(Optional)   | A string whose value is the database server name. The format for the name differs for different database systems. Specify an empty string "" if you connect to a local database and the server name is not required for connection. If you connect using ODBC (DBMS=ODBC) specify name of the desired ODBC profile as the <a href="#">server_name</a> |
| <a href="#">database_name</a><br>(Optional) | A string whose value is name of the database. Specify an empty string "" if the database name is not required for connection.   |
| <a href="#">user_id</a>                     | A string whose value is the name of the user to log on to the database  |
| <a href="#">password</a>                    | A string whose value is the password to use to log on to the database   |
| <a href="#">auto_commit</a>                 | A boolean whose value controls AutoCommit mode. Some databases support AutoCommit mode. Specify TRUE to turn AutoCommit on or specify FALSE otherwise. If DBMS does not support AutoCommit then this parameter is ignored.  |

**Return value:** None.

**Usage:** [DatabaseConnectEx](#) or [DatabaseConnect](#) statement must be executed before other database actions can be processed.

One job may have only one database connection open at a time. However, multiple jobs may have multiple database connections opened simultaneously.

All other database statements executed after [DatabaseConnectEx](#) are sent to the database connected in the same job.



**Important Note:** In some databases AutoCommit is required in order to execute DDL statements and create temporary tables. For example, this is required in MS SQL Server and Sybase SQL Server.

**See also:**

- DatabaseDisconnect
- Database Profiles
- DatabaseConnect

---

## DatabaseCopy

**Description:** Copies the data from the primary database buffer to the system clipboard.

**Syntax:** [DatabaseCopy return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows copied to the clipboard.

**Usage:** [DatabaseCopy](#) uses tab characters to separate columns and carriage return (CR), linefeed (LF) pairs to separate rows. The copied data can be easily pasted into a spreadsheet program or into other Windows applications that supports data format described above..

The data you want to copy can be previously retrieved from the database using [DatabaseRetrieve](#) statement or pasted from the system clipboard using [DatabasePaste](#) statement.

**See also:**

DatabasePaste  
Clipboard statements

---

## DatabaseDelete

**Description:** Deletes a row from the database buffer.

**Syntax:** [DatabaseDelete row](#)

| Argument            | Description                                     |
|---------------------|---|
| <a href="#">row</a> | A number identifying the row you want to delete |

**Return value:** None.

**Usage:** The row is not deleted from the database table until you call the [DatabaseUpdate](#) statement. After the [DatabaseUpdate](#) statement has successfully updated the database, then the storage associated with the row is cleared.

**See also:**

DatabaseRetrieve  
DatabaseRowCount  
DatabaseSetFilter  
DatabaseUpdate

## DatabaseDescribe

**Description:** Describes result set definition in the database buffer.

**Syntax:** [DatabaseDescribe return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">return</a> | A string variable that receives the returned value |

**Return value:** String. Returns current result set description.

**Usage:** This statement is provided for debugging purposes. The 24x7 Scheduler creates result set definition while executing [DatabaseSetSQLSelect](#) or [DatabaseRetrieve](#) statements. You can use [MessageBox](#) statement to display the returned result set definition.

**See also:**

- [MessageBox](#)
- [DatabaseRetrieve](#)
- [DatabaseSetSQLSelect](#)

---

## DatabaseDisconnect

**Description:** Disconnects from a database.

**Syntax:** [DatabaseDisconnect](#)

**Return value:** None.

**Usage:** [DatabaseDisconnect](#) statement has no arguments. This statement disconnects the 24x7 Scheduler from the database to which you previously connected using [DatabaseConnect](#) statement.

**See also:**

- [DatabaseConnect](#)
- [Database Profiles](#)

---

## DatabaseExecute

**Description:** Execute a SQL statement that does not produce a result set.

**Syntax:** [DatabaseExecute sql, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">sql</a>    | A string whose value is a valid SQL command that you want to send to the database. |
| <a href="#">return</a> | A numeric variable that receives the returned value                                |

**Return value:** Number. Returns the number of rows affected, if applicable.

**Usage:**

Refer to your database documentation for the correct SQL syntax. [DatabaseExecute](#) sends the specified SQL as is. Before you execute a SQL, be sure you have active database connection. You use the [DatabaseConnect](#) statement to establish a database connection.

**See also:**

[DatabaseRetrieve](#)

[DatabaseUpdate](#)

---

## DatabaseExport

**Description:** Exports data from the specified database table and in stores it in the specified file.

**Syntax:** [DatabaseExport table, file, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">table</a>  | A string whose value is the name of database table that you want to export             |
| <a href="#">file</a>   | A string whose value is the name of the file into which you want to save exported data |
| <a href="#">return</a> | A numeric variable that receives the returned value                                    |

**Return value:** Number. Returns the number of exported rows.

**See also:**

[DatabaseRetrieve](#)

[DatabaseImport](#)

[DatabaseSave](#)

---

## DatabaseGet

**Description:** Obtains the current value of a cell in the primary database buffer.

**Syntax:** [DatabaseGet row, column, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">row</a>    | A number whose value is the row number for the cell whose value you want to get    |
| <a href="#">column</a> | A number whose value is the column number for the cell whose value you want to get |
| <a href="#">return</a> | A string variable that receives the returned value                                 |

**Return value:** String. Returns the string representation of the data value at the [row](#) and [column](#) location. If the value is null, [DatabaseGet](#) returns the empty string ("").

**See also:**

[DatabaseSet](#)

[DatabaseUpdate](#)

---

## DatabaseImport

**Description:** Imports data from the specified file and inserts it into the specified database table.

**Syntax:** [DatabaseImport table, file, return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">table</a>  | A string whose value is the name of database table into which you want to import data |
| <a href="#">file</a>   | A string whose value is the name of the file that you want to import                  |
| <a href="#">return</a> | A numeric variable that receives the returned value                                   |

**Return value:** Number. Returns the number of imported rows.

**See also:**

[DatabaseRetrieve](#)

[DatabaseExport](#)

---

## DatabaseInsert

**Description:** Inserts an empty row into the database buffer.

**Syntax:** [DatabaseInsert row](#)

| Argument            | Description  |
|---------------------|--|
| <a href="#">row</a> | A number identifying the row before which you want to insert the <a href="#">row</a> . To insert a row at the end, specify <a href="#">0</a> |

**Return value:** None.

**Usage:** A newly inserted row is not marked as modified until data is entered in the row. You can use the [DatabaseSet](#) statement to enter data in the newly inserted row. The new modified row is not inserted into the database table until you call the [DatabaseUpdate](#) statement. After the [DatabaseUpdate](#) statement has successfully updated the database, the row state changes to unmodified.

**See also:**

[DatabaseRetrieve](#)  
[DatabaseRowCount](#)  
[DatabaseSet](#)  
[DatabaseUpdate](#)

---

## DatabasePaste

**Description:** Pastes the data from the system clipboard to the primary database buffer.

**Syntax:** [DatabasePaste return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows pasted from the clipboard.

**Usage:** The clipboard data is expected in ASCII text format with tab characters separating columns and carriage return (CR), linefeed (LF) pairs separating rows. The result set definition must exist in the database buffer before pasting data. Number of columns and their data types defined in the database buffer must match data in the clipboard. You can use either [DatabaseSetSQLSelect](#) or [DatabaseRetrieve](#) statement to create result set definition in the database buffer. You may want to call [DatabaseUpdate](#) statement after pasting data to save new data in the database.

**See also:**

[DatabaseCopy](#)  
[Clipboard statements](#)

---

## DatabasePipe

**Description:** Copies data from the source database to the destination database as specified by the SQL query and update method.

**Syntax:** [DatabasePipe source, destination, query, table, method, return](#)

| Argument                    | Description   |
|-----------------------------|---|
| <a href="#">source</a>      | A string whose value is the profile name of the source database   |
| <a href="#">destination</a> | A string whose value is the profile name of the destination database  |
| <a href="#">query</a>       | A string whose value a valid SQL SELECT statement that is used to retrieve data from the source database  |
| <a href="#">table</a>       | A string whose value is the name of database table in the <a href="#">destination</a> database into which you want to copy data from the <a href="#">source</a> database  |
| <a href="#">method</a>      | A string constant whose value specifies how you want to update the destination <a href="#">table</a> . Values are: <ul style="list-style-type: none"><li>• <b>"INSERT"</b> — Execute DELETE then INSERT SQL statements for each copied row</li><li>• <b>"UPDATE"</b> — Execute UPDATE SQL statement for each copied row</li></ul> |
| <a href="#">return</a>      | A numeric variable that receives the returned value   |

**Return value:** Number. Returns the number of transferred rows.

**Usage:** Before the 24x7 Scheduler executes the pipeline, it connects to the source and destination databases. This does not affect active database connection if any. When the 24x7 Scheduler executes the pipeline, the piped data is committed every 100 rows. The data is also committed when the pipeline finishes. If a database error occurs, last not committed yet data portion is rolled back.

**Caution:**

Special considerations for the "UPDATE" method:

- For each row to be updated you should have at least one matching row in the destination table. Otherwise a database error will occur. The match is based on the primary key values.
- The primary key must be defined for the destination table. [DatabasePipe](#) uses primary key definition when constructing the UPDATE SQL statements.

Special considerations for the "INSERT" method:

- [DatabasePipe](#) generates and executes the DELETE SQL statement for each new row. Then it generates the INSERT SQL statements for each new row. This method allows avoiding duplicate records inserted in the destination table. However, if there is a foreign key of type "DELETE CASCADE" defined for the destination table, the database will enforce such referential integrity and will delete all referencing records from the child table(s).
- The primary key must be defined for the destination table. [DatabasePipe](#) uses primary key definition when constructing the DELETE SQL statements.

**See also:**

DatabaseRetrieve  
DatabaseExport  
DatabaseImport  
DatabaseSave

## DatabaseRetrieve

**Description:** Retrieves data from the database using the specified SQL query.

**Syntax:** [DatabaseRetrieve query, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">query</a>  | A string whose value is a valid SQL SELECT statement that is used to retrieve data |
| <a href="#">return</a> | A numeric variable that receives the number of rows retrieved                      |

**Return value:** Number. Returns the number of retrieved rows.

**Usage:** After rows are retrieved, the database buffer's filter and sort options are applied. Therefore, any retrieved rows that don't meet the filter criteria are not included in the returned data and number of rows. The retrieved data remains in the database buffer until new retrieval. A new call to the [DatabaseFilter](#) and/or [DatabaseSort](#) statements will affect data in the primary database buffer.

Before you retrieve rows, be sure you have active database connection. You use the [DatabaseConnect](#) statement to establish a database connection.

**See also:**

[DatabaseExport](#)  
[DatabaseConnect](#)

---

## DatabaseRowCount

**Description:** Obtains the number of rows that are currently available in the primary database buffer. If there is a filter set on the database buffer, the [DatabaseRowCount](#) statement checks only rows that have been left.

**Syntax:** [DatabaseRowCount return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A numeric variable that receives the number of rows retrieved |

**Return value:** Number. Returns the number of rows that are currently available in the primary database buffer, [0](#) if no rows are currently available, and [-1](#) if an error occurs.

**Usage:** The number of currently available rows equals the total number of rows retrieved plus any pasted rows minus any rows that have been filtered out.

**See also:**

[DatabaseSetFilter](#)

## DatabaseSave

**Description:** Saves the contents of the primary database buffer as an external file in the specified format.

**Syntax:** [DatabaseSave file, format, return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">file</a>   | A string whose value is the name of the file in which to save the contents. If you specify an empty string (""), the 24x7 Scheduler prompts for the file name   |
| <a href="#">format</a> | A string constant specifying the format in which to save the contents of the primary database buffer. Values are:<br><br>"DBF" — dBASE-III format<br><br>"DIF" — Data Interchange Format<br><br>"XLS" — Microsoft Excel 5 format<br><br>"HTML" or "HTM" — table in HTML format<br><br>"SQL" — SQL syntax with CREATE TABLE statement following by INSERT statements<br><br>"TXT" — Text format with tab-separated columns and return at the end of each row<br><br>"CSV" — Text format with tab-separated columns and return at the end of each row<br><br>"WKS" — Lotus 1-2-3 format |
| <a href="#">return</a> | A numeric variable that receives the returned value   |

**Return value:** Number. Returns the number of rows saved in the specified [file](#).

**Usage:** The number of rows currently available for saving equals the total number of rows retrieved plus any pasted rows minus any rows that have been filtered out. If you specify an empty string for the [file](#) name, the 24x7 Scheduler displays the Save Rows As dialog. Execution of the script will not continue until you either enter missing [file](#) name interactively or cancel the dialog.

**See also:**

- DatabaseExport
- DatabaseRetrieve
- DatabaseSetFilter

---

## DatabaseSet

**Description:** Changes the value of a cell in the primary database buffer to the specified value.

**Syntax:** [DatabaseSet row, column, value](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">row</a>    | A number whose value is the row number for the cell whose value you want to change  |
| <a href="#">column</a> | A number whose value is the column number for the cell whose value you want to change   |
| <a href="#">value</a>  | A string whose value is the string representation of the new value to which you want to set the data at the <a href="#">row</a> and <a href="#">column</a> location |

**Return value:** None.

**Usage:** The value is not changed in the database table until you call the [DatabaseUpdate](#) statement. [DatabaseSet](#) statement automatically converts the specified [value](#) to the appropriate data type that matches data type of the database table that you want to update.

**See also:**

[DatabaseGet](#)

[DatabaseUpdate](#)

---

## DatabaseSetFilter

**Description:** Changes filter criteria for the database buffer. Rows that do not meet the filter criteria are moved to the filter buffer.

**Syntax:** [DatabaseSetFilter filter, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">filter</a> | A string whose value is a boolean expression that you want to use as the filter criteria. The expression includes column names or numbers. A column number must be preceded by a pound sign (#). |

**Return value:** None.

**Usage:** The database buffer can have [filter](#) criteria specified as part of its definition. After data is retrieved, rows that don't meet the criteria are immediately transferred from the primary database buffer to the filter buffer.

The [DatabaseSetFilter](#) statement replaces the [filter](#) criteria defined for the database buffer, if any, with a new set of criteria. The new [filter](#) is applied immediately.

The [filter](#) expression consists of columns, functions, operators, and values against which column values are compared. Boolean expressions can be connected with [AND](#) and [OR](#) logical operators. You can also use [NOT](#), the negation operator. Use parentheses to control the order of evaluation.

Sample expressions are:

```
item_id > 5
NOT item_id = 5
(NOT item_id = 5) AND customer > "Mabson"
item_id > 5 AND customer = "Smith"
left(customer, 5) = "Smith"
#1 > 5 AND #2 = "Smith"
```

The [filter](#) expression is a string and does not contain variables. However, you can build the string at during execution using the values of variables in the script. Within the filter string, values that are strings must be enclosed in quotation marks (see the examples).

If the [filter](#) expression contains numbers, the database buffer expects the numbers in U.S. format. Be aware that when converting numbers to string the [String](#) statement formats numbers using the current system settings. If you use it to build the filter expression, use the [Format](#) statement to specify the desired format.

To remove a [filter](#), call [DatabaseSetFilter](#) with the empty string (""). The rows in the filter buffer will be restored to the primary database buffer after the rows that were already in the primary buffer.

**See also:**

[DatabaseSetSort](#)

[Operators in Filter and Sort Expressions](#)

[Functions in Filter and Sort Expressions by Category](#)

---

## DatabaseSetSort

**Description:** Changes sort criteria for the database buffer.

**Syntax:** [DatabaseSetSort](#) [sort](#), [return](#)

| Argument             | Description   |
|----------------------|---|
| <a href="#">sort</a> | A string whose value is an expression that you want to use as the sort criteria. The expression includes column names or numbers. A column number must be preceded by a pound sign (#). |

**Return value:** None.

**Usage:** The database buffer can have [sort](#) criteria specified as part of its definition. After data is retrieved, the 24x7 Scheduler performs immediate sorting according to the current [sort](#) criteria.

The [DatabaseSetSort](#) statement replaces the [sort](#) criteria defined for the database buffer, if any, with a new set of criteria. The new [sort](#) is applied immediately.

The [sort](#) expression may include columns and functions.

The [sort](#) criteria for a column has one of the forms shown in the following table, depending on whether you specify the column by name or number. Order is either A for ascending or D for descending order. You can specify secondary sorting by specifying criteria for additional columns in the format string. Separate each column specification with a comma.

| Syntax for sort order | Examples                                  |
|-----------------------|---|
| columnname order      | "emp_lname A"<br>"emp_lname A, dept_id D" |
| # columnnumber order  | "#3 A"                                    |

In place of a column in the [sort](#) criteria you can use an expressions.

Sample expressions are:

```
"(item_id * 5) A"  
if(status = 'A', current_rate, prev_rate * 01) A  
left(#1, 5) = D, #3 A
```

The [sort](#) expression is a string and does not contain variables. However, you can build the string at during execution using the values of variables in the script. Within the [sort](#) string, values that are strings must be enclosed in quotation marks (see the examples).

If the [sort](#) expression contains numbers, the database buffer expects the numbers in U.S. format. Be aware that when converting numbers to string the [String](#) statement formats numbers using the current system settings. If you use it to build the filter expression, use the [Format](#) statement to specify the desired format.

To remove the database buffer sorting, call [DatabaseSetSort](#) with the empty string ("" ). However the empty string ("" ) does not restore original sort order made in the database at during retrieval of data.

**See also:**

[DatabaseSetFilter](#)

[Operators in Filter and Sort Expressions](#)

[Functions in Filter and Sort Expressions by Category](#)

---

## DatabaseSetSQLSelect

**Description:** Specifies the new SQL SELECT statement for the database buffer. The result set definition in the database buffer is updated to match the new SQL.

**Syntax:** [DatabaseSetSQLSelect sql](#)

| Argument            | Description   |
|---------------------|---|
| <a href="#">sql</a> | A string whose value is the new SQL SELECT statement that you want to use for the database operations |

**Return value:** None.

**Usage:** The [DatabaseSetSQLSelect](#) statement destroys data in the database buffer, if any, then creates the new result set definition based on supplied SELECT statement. Any existing filter and sort criteria are also reset.

**See also:**

[DatabaseRetrieve](#)

---

## DatabaseUpdate

**Description:** Updates the database with the changes made.

**Syntax:** [DatabaseUpdate method, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">method</a> | A string constant whose value specifies how you want to update the destination table. Values are: <ul style="list-style-type: none"><li>• <a href="#">"INSERT"</a> — Execute DELETE then INSERT SQL statements for each modified row</li></ul> |

|               |  |
|---------------|--|
|               | <ul style="list-style-type: none"><li>• <b>"UPDATE"</b> — Execute UPDATE SQL statement for each modified row</li></ul> |
| <b>return</b> | A numeric variable that receives the returned value  |

**Return value:** Number. Returns the number of newly updated/inserted/deleted rows.

**Usage:** The [DatabaseUpdate](#) statement generates UPDATE, INSERT, and DELETE SQL statements for all new or changed rows in the database buffer including rows in the filter buffer, if any. The [DatabaseUpdate](#) always generates the INSERT SQL statement for all new rows. Depends on the specified update [method](#), [DatabaseUpdate](#) determines for each modified row, whether the row can be updated in place or whether the row has to be deleted and reinserted. The database changes are committed after the last row is updated. If a database error occurs, all database changes are rolled back.

**Caution:**

Special considerations for the "UPDATE" method:

- When there are multiple rows modified in the database buffer and you have switched keys or rows, updating in place may fail due to DBMS duplicate restrictions.
- The primary key must be defined for the destination table. [DatabaseUpdate](#) uses primary key definition when constructing UPDATE SQL statements.

Special considerations for the "INSERT" method:

- [DatabaseUpdate](#) generates and executes the DELETE SQL statement for each row that needs to be updated. Then it generates the INSERT SQL statements that reinserts each modified row in the database. This method allows avoiding duplicate records inserted in the destination table. However, if there is a foreign key of type "DELETE CASCADE" defined for the destination table, the database will enforce such referential integrity and will delete all referencing records from the child table(s).
- The primary key must be defined for the destination table. [DatabaseUpdate](#) uses primary key definition when constructing DELETE SQL statements.

**See also:**

DatabaseRetrieve  
DatabaseImport  
DatabasePipe  
DatabasePaste  
DatabaseDelete

---

## Date and time statements

---

### Date

**Description:** Converts a string whose value is a valid date to a value of data type date.

**Syntax:** [Date](#) ( [string](#), [result](#) )

| Argument               | Description  |
|------------------------|--|
| <a href="#">String</a> | A string containing a valid date (such as Jan 1, 1998, or 12-31-99) that you want returned as a date |

|        |  |
|--------|--|
| Result | A date variable that receives the returned value |
|--------|--|

**Return value:** Date. Returns the date in [string](#) as a date. If [string](#) does not contain a valid date, Date returns 1900-01-01.

**Usage:** The value of the [string](#) must be a valid date.



**Note:**

Valid dates in strings can include any combination of day (1 to 31), month (1 to 12 or the name or abbreviation of a month), and year (2 or 4 digits). 24x7 Scheduler assumes a 4-digit number is a year. Leading zeros are optional for month and day. The month, whether a name, an abbreviation, or a number, must be in the month location specified in the system setting for a date's format. If you do not know the system setting, use the standard data type date format "yyyy-mm-dd".

A four-digit number is assumed to be a year. If the year is two digits, then 24x7 Scheduler chooses the century, as follows: If the year is between 00 and 49, program assumes 20 as the first two digits; if it is between 50 and 99, the 24x7 Scheduler assumes 19. If your data includes dates before 1950, such as birth dates, always specify a four-digit year so that the 24x7 Scheduler interprets the date as intended.

The 24x7 Scheduler handles years from 1000 to 3000 inclusive.

**See also:**

DateTime statement

MakeDate statement

---

## DateTime

**Description:** Combines a date and a time value into a DateTime value.

**Syntax:** [DateTime](#) ( [date](#), [time](#), [result](#) )

| Argument               | Description  |
|------------------------|--|
| <a href="#">date</a>   | A valid date (such as Jan 1, 1998, or 12-31-99) that you want included in the value returned by DateTime   |
| <a href="#">time</a>   | A valid time (such as 8AM or 10:25:23) that you want included in the value returned by DateTime. If you include a time, only the hour portion is required. If you omit the minutes, or seconds, they are assumed to be 0s. If you omit AM or PM, the hour is determined according to the 24-hour clock |
| <a href="#">result</a> | A datetime variable that receives the returned value   |

**Return value:** DateTime. Returns a DateTime value based on the values in [date](#) and [time](#).

**Usage:** For information on valid dates, see Date statement.

## DateAdd

**Description:** Obtains the date that occurs a specified number of days after or before another date.

**Syntax:** `DateAdd ( date, n, result )`

| Argument            | Description                                      |
|---------------------|--|
| <code>date</code>   | A date value                                     |
| <code>n</code>      | An number indicating the number of days          |
| <code>result</code> | A date variable that receives the returned value |

**Return value:** Date. Returns the date that occurs `n` days after `date` if `n` is greater than 0. Returns the date that occurs `n` days before `date` if `n` is less than 0.

**See also:**

TimeAdd

DateTimeAdd

---

## DateDiff

**Description:** Determines the number of days between two specified dates..

**Syntax:** `DateDiff ( date1, date2, result )`

| Argument            | Description  |
|---------------------|--|
| <code>date1</code>  | A date value that is the start date of the interval being measured |
| <code>date2</code>  | A date value that is the end date of the interval                  |
| <code>result</code> | A numeric variable that receives the returned value                |

**Return value:** Number. Returns the number of days `date2` occurs after `date1`. If `date2` occurs before `date1`, DaysDiff returns a negative number.

**See also:**

DateTimeDiff

TimeDiff

---

## DateTimeAdd

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** `DateTimeAdd ( datetime, n, result )`

| Argument              | Description  |
|-----------------------|--|
| <code>datetime</code> | A datetime value                                     |
| <code>n</code>        | A long number of seconds                             |
| <code>result</code>   | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns the time that occurs `n` seconds after `datetime` if `n` is greater than 0. Returns the time that occurs `n` seconds before `datetime` if `n` is less than 0.

**See also:**

TimeAdd  
DateAdd

---

## DateTimeDiff

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** `DateTimeDiff ( datetime1, datetime2, result )`

| Argument               | Description  |
|------------------------|--|
| <code>datetime1</code> | A datetime value that is the start time of the interval being measured |
| <code>datetime2</code> | A datetime value that is the end time of the interval                  |
| <code>result</code>    | A numeric variable that receives the returned value                    |

**Return value:** Number. Returns the number of seconds `datetime2` occurs after `datetime1`. If `datetime2` occurs before `datetime1`, TimeDiff returns a negative number.

**See also:**

DateDiff  
TimeDiff

---

## DateTimePart

**Description:** Extracts the number containing the specified part of a given datetime, date, or string value.

**Syntax:** `DateTimePart ( datetime, part, result )`

| Argument              | Description  |
|-----------------------|--|
| <code>datetime</code> | A datetime (optionally date or string) value from which you want to extract specified interval of time |
| <code>part</code>     | A string value that specifies the interval of time you want to return                                  |
| <code>result</code>   | A numeric variable that receives the returned value  |

**Return value:** Number. Returns the interval of time extracted from `datetime` value.

**Settings:** The interval argument has these settings:

| Setting         | Description  |
|-----------------|--------------|
| <code>dd</code> | day of month |
| <code>dw</code> | day of week  |
| <code>mm</code> | Month        |
| <code>yy</code> | Year         |
| <code>hh</code> | Hour         |
| <code>mi</code> | Minute       |
| <code>ss</code> | Second       |
| <code>qq</code> | Quarter      |

---

## DayName

**Description:** Determines the day of the week in a date value and returns the weekday's name.

**Syntax:** `DayName (date, result )`

| Argument            | Description  |
|---------------------|--|
| <code>date</code>   | The date for which you want the name of the day    |
| <code>result</code> | A string variable that receives the returned value |

**Return value:** String. Returns a string whose value is the name of the weekday (Sunday, Monday, and so on) for `date`.

**See also:**  
DayName

## DayNumber

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** `DayNumber (date, result )`

| Argument            | Description  |
|---------------------|--|
| <code>date</code>   | The date from which you want the number of the day of the week |
| <code>result</code> | A numeric variable that receives the returned value            |

**Return value:** Number. Returns an integer (1-7) representing the day of the week of `date`. Sunday is day 1, Monday is day 2, and so on.

**See also:**

DayNumber  
Macro-parameters

---

## MakeDate

**Description:** Combines numbers representing the year, month, and day into a date value.

**Syntax:** `MakeDate ( year, month, day, result )`

| Argument            | Description  |
|---------------------|--|
| <code>year</code>   | The 4-digit year (1000 to 3000) of the date                  |
| <code>month</code>  | The 1- or 2-digit number for the month (1 to 12) of the year |
| <code>day</code>    | The 1- or 2-digit number for the day (1 to 31) of the month  |
| <code>result</code> | A date variable that receives the returned value             |

**Return value:** Date. Returns the date specified by the numbers for `year`, `month`, and `day` as a date data type. If any value is invalid (out of the range of values for dates), MakeDate returns 1900-01-01.

## MakeDateTime

**Description:** Combines numbers representing the year, month, day, hour, minute, and second into a datetime value.

**Syntax:** `MakeDateTime ( year, month, day, hour, minute, second, result )`

| Argument            | Description  |
|---------------------|--|
| <code>year</code>   | The 4-digit year (1000 to 3000) of the date                  |
| <code>month</code>  | The 1- or 2-digit number for the month (1 to 12) of the year |
| <code>day</code>    | The 1- or 2-digit number for the day (1 to 31) of the month  |
| <code>hour</code>   | The number for the hour (00 to 23) of the time               |
| <code>minute</code> | The number for the minutes (00 to 59) of the time            |
| <code>second</code> | The number for the seconds (0 to 59) of the time             |
| <code>result</code> | A datetime variable that receives the returned value         |

**Return value:** DateTime. Returns the datetime value based on the values in `year`, `month`, `day`, `hour`, `minute`, and `second` arguments.

---

## MakeTime

**Description:** Combines numbers representing hours, minutes, and seconds into a time value.

**Syntax:** `MakeTime ( hour, minute, second, result )`

| Argument            | Description                                       |
|---------------------|---|
| <code>hour</code>   | The number for the hour (00 to 23) of the time    |
| <code>minute</code> | The number for the minutes (00 to 59) of the time |
| <code>second</code> | The number for the seconds (0 to 59) of the time  |
| <code>result</code> | A time variable that receives the returned value  |

**Return value:** Time. Returns the time as a time data type and 00:00:00 if the value in any argument is not valid (out of the specified range of values).

## Now

**Description:** Obtains the system time

**Syntax:** `Now ( result )`

| Argument            | Description                                      |
|---------------------|--|
| <code>result</code> | A time variable that receives the returned value |

**Return value:** Time. Returns the current system time.

**See also:**

- Today
- Timer
- Macro-parameters

---

## Timer

**Description:** Reports the amount of CPU time that has elapsed since specified `starttime`.

**Syntax:** `Timer starttime, result`

| Argument               | Description  |
|------------------------|--|
| <code>starttime</code> | A number whose value is start of the time interval expressed in milliseconds |
| <code>result</code>    | A number variable that receives the returned value.                          |

**Return value:** Number. Returns number of milliseconds elapsed since `starttime`. If you specify 0 for `starttime` the Timer statement returns number of milliseconds elapsed since 24x7 Scheduler startup.

**See also:**

- Today
- Now
- Macro-parameters

---

## Time

**Description:** Converts a string to a time data type.

**Syntax:** `Time ( string, result )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | A string containing a valid time (such as 8AM or 10:25) that you want returned as a time data type. Only the hour is required; you do not have to include the minutes, or seconds of the time or AM or PM. The default value for minutes and seconds is 00 . AM or PM is determined automatically |
| <code>result</code> | A time variable that receives the returned value  |

**Return value:** Time. Returns the time in `string` as a time data type. If `string` does not contain a valid time, Time returns 00:00:00.

---

## TimeAdd

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** `TimeAdd ( time, n, result )`

| Argument            | Description                                      |
|---------------------|--|
| <code>time</code>   | A time value                                     |
| <code>n</code>      | A long number of seconds                         |
| <code>result</code> | A time variable that receives the returned value |

**Return value:** Time. Returns the time that occurs `n` seconds after `time` if `n` is greater than 0. Returns the time that occurs `n` seconds before `time` if `n` is less than 0. The maximum return value is 23:59:59.

**See also:**

DateAdd

DateTimeAdd

---

## TimeDiff

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** `TimeDiff ( time1, time2, result )`

| Argument           | Description  |
|--------------------|--|
| <code>time1</code> | A time value that is the start time of the interval being measured |

|                        |   |
|------------------------|---|
| <a href="#">time2</a>  | A time value that is the end time of the interval   |
| <a href="#">result</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of seconds [time2](#) occurs after [time1](#). If [time2](#) occurs before [time1](#), TimeDiff returns a negative number.

**See also:**

DateTimeDiff  
DateDiff

---

## Today

**Description:** Obtains the system date.

**Syntax:** [Today](#) ( [result](#) )

| Argument               | Description                                      |
|------------------------|--|
| <a href="#">result</a> | A date variable that receives the returned value |

**Return value:** Date. Returns the current system date.

**See also:**

Now  
Macro-parameters

---

## DDE statements

---

### DDEClose

**Description:** Closes a channel to a DDE server application that was opened by calling the [DDEOpen](#) statement.

**Syntax:** [DDEClose](#) [channel](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">channel</a> | A number whose value is the channel number previously returned by the <a href="#">DDEOpen</a> statement that started the DDE conversation |

**Return value:** None.

**Usage:** Use [DDEClose](#) to close a channel to a DDE server application that was opened by calling the [DDEOpen](#) statement .

**See also:**

DDEOpen

---

## DDEExecute

**Description:** Asks a DDE server application to execute the specified command.

**Syntax:** [DDEExecute channel, command](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">channel</a> | A number whose value is the channel number previously returned by the <a href="#">DDEOpen</a> statement that started the DDE conversation                                  |
| <a href="#">command</a> | A string whose value is the command you want a DDE server application to execute. The format of the command depends on the DDE application you want to execute the command |

**Return value:** None.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the [Run](#) statement to start the application if necessary. Before using this statement, call [DDEOpen](#) to establish a DDE channel.

**See also:**

DDEOpen

---

## DDEGetData

**Description:** Asks a DDE server application to provide data and stores that data in the specified variable.

**Syntax:** [DDEGetData channel, location, return](#)

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">channel</a>  | A number whose value is the channel number previously returned by the <a href="#">DDEOpen</a> statement that started the DDE conversation |
| <a href="#">location</a> | A string whose value is the location of the data you want returned. The format of the location depends on the DDE                         |

|                        |  |
|------------------------|--|
|                        | application that will receive the request          |
| <a href="#">return</a> | A string variable that receives the returned value |

**Return value:** String.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the [Run](#) statement to start the application if necessary. Before using this statement, call [DDEOpen](#) to establish a DDE channel.

**See also:**

[DDEOpen](#)

---

## DDEOpen

**Description:** Opens a channel to a DDE server application.

**Syntax:** [DDEOpen](#) *application*, *topic*, *return*

| Argument                    | Description   |
|-----------------------------|---|
| <a href="#">application</a> | A string specifying the DDE name of the DDE server application  |
| <a href="#">topic</a>       | A string identifying the data or the instance of the application you want to use (for example, in Microsoft Excel, the topic name could be <a href="#">System</a> or the name of an open spreadsheet) |
| <a href="#">return</a>      | A numeric variable that receives the returned value   |

**Return value:** Number. Returns the handle to the channel if it succeeds. If an error occurs, [DDEOpen](#) returns a negative number.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the [Run](#) statement to start the application if necessary.

**See also:**

[DDEClose](#)

---

## DDESetData

**Description:** Asks a DDE server application to accept data and store it in the specified location.

**Syntax:** [DDESetData](#) *channel*, *location*, *data*

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">channel</a>  | A number whose value is the channel number previously returned by the <a href="#">DDEOpen</a> statement that started the DDE conversation                   |
| <a href="#">location</a> | A string whose value is the location of the data you want returned. The format of the location depends on the DDE application that will receive the request |
| <a href="#">data</a>     | A string whose value you want to send to the DDE server application   |

**Return value:** None.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the [Run](#) statement to start the application if necessary. Before using this statement, call [DDEOpen](#) to establish a DDE channel.

**See also:**

[DDEOpen](#)

---

## File statements

---

### Dir

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** [Dir file\\_mask, return](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">File_mask</a> | A string whose value is the DOS file mask that you want to use to list file. <a href="#">File_mask</a> can contain wildcard characters (* and ?). <a href="#">File_mask</a> can contain full or partial file path. |
| <a href="#">Return</a>    | A string variable that receives the returned value.  |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The [Dir](#) statement is equivalent to DOS *dir* command.

To get the list of all files use \*.\* file mask. If you don't include file path to the [file\\_mask](#) then [Dir](#) statement returns files from the DOS current directory.

**See also:**

[CD](#)  
[FileSearchEx](#)  
[FileFindFirst](#)  
[FTPDDir](#)  
[RemoteDir](#)

---

## DirWaitForUpdate

**Description:** Suspends job execution and then enters an efficient wait state until either a change occurs in the monitored directory or the statement times out.

**Syntax:** `DirWaitForUpdate dir, timeout, return_status`

| Argument                   | Description   |
|----------------------------|---|
| <code>dir</code>           | A string whose value is the full name of the directory that you want to monitor.  |
| <code>timeout</code>       | A number whose value is the maximum time interval within which you expect a new change to occur in the specified <code>dir</code> . Use <code>0</code> timeout to allow infinite waiting. |
| <code>return_status</code> | A boolean variable that receives the returned status value.   |

**Return value:** Boolean. If the `DirWaitForUpdate` statement succeeds, the `return_status` is `True`, otherwise the `return_status` value is `False`. Make sure to specify a valid directory name. Invalid names will cause the statement to return `False`.

**See also:**

- Dir
- Wait
- FileTime
- FileSize

---

## EOF

**Description:** Checks whether the end of a file (EOF) opened for read operation has been reached.

**Syntax:** `EOF filenum, return`

| Argument             | Description   |
|----------------------|---|
| <code>Filenum</code> | The file number previously returned by FileOpen statement |
| <code>return</code>  | A boolean variable that receives the returned value       |

**Return value:** Boolean. Returns TRUE when the end of a file has been reached. The `EOF` statement returns FALSE until the end of the file has been reached.

**Usage:** Use [EOF](#) to avoid the error generated by attempting to read past the end of a file. With files opened for write operations, [EOF](#) always returns TRUE.

**See also:**

FileOpen  
FileSetPos  
FileGetPos  
FileSize

---

## FileExists

**Description:** Reports whether the specified file exists.

**Syntax:** [FileExists](#) file, return

| Argument               | Description   |
|------------------------|---|
| <a href="#">file</a>   | A string whose value is the name of a file          |
| <a href="#">return</a> | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the file exists, FALSE if it does not exist.

**Usage:** If [file](#) is locked by another application, causing a sharing violation, [FileExists](#) also returns FALSE. [File](#) name may include wildcard characters (\* and ?).

**See also:**

NotFileExists

---

## FileClose

**Description:** Closes the file associated with the specified file number. The file number was assigned to the file with the FileOpen statement.

**Syntax:** [FileClose](#) filenum

| Argument                | Description   |
|-------------------------|---|
| <a href="#">filenum</a> | The number assigned to the file you want to close. The <a href="#">FileOpen</a> statements returns the file number when it opens the file |

**Return value:** None

**See also:**

FileOpen

## FileCopy

**Description:** Copies an existing file to a new file.

**Syntax:** `FileCopy source, destination`

| Argument                 | Description   |
|--------------------------|---|
| <code>source</code>      | A string whose value is the name of an existing file you want to copy                                 |
| <code>destination</code> | A string whose value is the name of a new file to which you want to copy the <code>source</code> file |

**Return value:** None

**Usage:** If the `destination` file already exists, the statement overwrites the existing file.

**See also:**

- FileCopyEx
- FileMove
- FileTransfer

---

## FileCopyEx

**Description:** Copies existing files to a new location.

**Syntax:** `FileCopyEx source, destination, return`

| Argument                 | Description   |
|--------------------------|---|
| <code>source</code>      | A string whose value is the file mask describing existing files that you want to copy. <code>Source</code> file mask can contain wildcard characters (* and ?). |
| <code>destination</code> | A string whose value is the directory name to which you want to copy the <code>source</code> files.   |
| <code>return</code>      | A numeric variable that receives the returned value.  |

**Return value:** Number of files copied. If no files were found to satisfy `source` file mask, the return value is 0.

**Usage:** If any of the source files already exists in the `destination`, the statement overwrites the existing file.

**See also:**

- FileCopy
- FileMoveEx

---

## FileTransfer

**Description:** Copies an existing file from the local drive to a new file on the remote computer.

**Syntax:** `FileTransfer host, source, destination`

| Argument                 | Description   |
|--------------------------|---|
| <code>host</code>        | A string whose value is the name of a Remote Agent that is running on the <code>host</code> computer  |
| <code>source</code>      | A string whose value is the name of an existing file you want to transfer to the <code>host</code> computer   |
| <code>destination</code> | A string whose value is the name of a new file which you want to create on the <code>host</code> computer as a copy of the <code>source</code> file |

**Return value:** None

**Usage:** If the `destination` file already exists, the statement overwrites the existing file. The `host` name must match the name of an existing Remote Agent profile on the computer initiating the transfer. The transfer time might be lengthily for large files and slow networks. To reduce network traffic and transmission time 24x7 Scheduler compresses `source` file before sending it to the `host`. The Remote Agent decompresses received data and stores them in the `destination` file.



### Tips:

- You can transfer multiple files at once. Use comma to separate multiple files in both `source` and `destination` parameters. Make sure to specify matching number of files for the `source` and `destination` parameters. Sending multiple files in one `FileTransfer` batch is faster than sending source files one by one using multiple `FileTransfer` statements.
- You must have sufficient free disk space on both sending and host computers for the temporary files used in compression/decompressions routines.

### See also:

FileMove

FileCopy

---

## FileMove

**Description:** Moves an existing file to a new file.

**Syntax:** `FileMove source, destination`

| Argument                    | Description  |
|-----------------------------|--|
| <a href="#">source</a>      | A string whose value is the name of an existing file you want to move                                    |
| <a href="#">destination</a> | A string whose value is the name of a new file to which you want to move the <a href="#">source</a> file |

**Return value:** None

**Usage:** If the [destination](#) file already exists, the statement overwrites the existing file. [FileMove](#) first checks the [destination](#) file and deletes it when this file exists. Then [FileMove](#) copies the [source](#) file to the new file. If the copy operation succeeds, [FileMove](#) deletes the [source](#) file.

**See also:**

FileCopy  
FileMoveEx

---

## FileMoveEx

**Description:** Moves existing files to a new location.

**Syntax:** [FileMoveEx](#) [source](#), [destination](#), [return](#)

| Argument                    | Description  |
|-----------------------------|--|
| <a href="#">source</a>      | A string whose value is the file mask describing existing files that you want to move. <a href="#">Source</a> file mask can contain wildcard characters (* and ?). |
| <a href="#">destination</a> | A string whose value is the directory name to which you want to move the <a href="#">source</a> files.   |
| <a href="#">return</a>      | A numeric variable that receives the returned value.   |

**Return value:** Number of files moved. If no files were found to satisfy [source](#) file mask, the return value is 0.

**Usage:** If any of the source files already exists in the [destination](#), the statement overwrites the existing file.

**See also:**

FileMove  
FileCopyEx  
FileTransfer

---

## FileDate

**Description:** Reports the date that a file was last modified.

**Syntax:** [FileDate](#) [file](#), [return](#)

| Argument               | Description                                      |
|------------------------|--|
| <a href="#">file</a>   | A string whose value is the name of a file       |
| <a href="#">return</a> | A date variable that receives the returned value |

**Return value:** Date. Returns the date that a file was last modified..

**See also:**

FileTime

MakeDateTime

---

## FileTime

**Description:** Reports the time that a file was last modified.

**Syntax:** [FileTime](#) [file](#), [return](#)

| Argument               | Description                                      |
|------------------------|--|
| <a href="#">file</a>   | A string whose value is the name of a file       |
| <a href="#">return</a> | A time variable that receives the returned value |

**Return value:** Time. Returns the time that a file was last modified.

**See also:**

FileDate

MakeDateTime

---

## FileDelete

**Description:** Deletes the specified file.

**Syntax:** [FileDelete](#) [file](#)

| Argument             | Description  |
|----------------------|--|
| <a href="#">file</a> | A string whose value is the name of a file you want delete |

**Return value:** None

**See also:**

FileDeleteEx

---

## FileDeleteEx

**Description:** Deletes existing files from the specified location.**Syntax:** [FileDeleteEx mask, return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">mask</a>   | A string whose value is the file mask describing existing files that you want to delete. File <a href="#">mask</a> can contain can contain wildcard characters (* and ?). |
| <a href="#">return</a> | A numeric variable that receives the returned value.  |

**Return value:** Number of files deleted. If no files were found to satisfy file [mask](#), the return value is [0](#).**See also:**

FileDelete

FileMoveEx

---

## FileFindFirst

**Description:** Searches a directory for a file whose name matches the specified filename. [FileFindFirst](#) examines subdirectory names as well as filenames.**Syntax:** [FileFindFirst filename, firstfile, return](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">filename</a>  | A string whose value specifies a valid directory or path and filename, which can contain wildcard characters (* and ?).  |
| <a href="#">firstfile</a> | A string variable that receives the name of a file whose name matches the specified <a href="#">filename</a>   |
| <a href="#">return</a>    | A boolean variable that receives the success of the operation. If the search was successful <a href="#">return</a> receives TRUE, otherwise it receives FALSE. |

**Return value:** This statements returns two values: string and boolean. If the statement succeeds, the string variable [firstfile](#) receives the name of the first found file.**Usage:** If the statement succeeds, you can use the [FileFindNext](#) statement to search for other files that match the same pattern. You may want to call [FileFindNext](#) in a loop in order to find all files that match the same pattern.

**See also:**

FileFindNext  
Dir  
FileExists  
LoopWhile  
LoopUntil

---

## FileFindNext

**Description:** Searches a directory for a file whose name matches the specified filename. [FileFindNext](#) examines subdirectory names as well as filenames.

**Syntax:** [FileFindNext](#) [nextfile](#), [return](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">nextfile</a> | A string variable that receives the name of a file whose name matches the mask specified in the previous call to the <a href="#">FindFileFirst</a> statement   |
| <a href="#">return</a>   | A boolean variable that receives the success of the operation. If the search was successful <a href="#">return</a> receives TRUE, otherwise it receives FALSE. |

**Return value:** This statements returns two values: string and boolean. If the statement succeeds, the string variable [nextfile](#) receives the name of the next found file.

**Usage:** You use the [FileFindNext](#) statement to continue a file search from a previous call to the [FindFileFirst](#) statement. You may want to call [FileFindNext](#) in a loop in order to find all files that match the same pattern.

**See also:**

FileFindFirst  
Dir  
FileExists  
LoopWhile  
LoopUntil

---

## FileSearchEx

**Description:** Searches for files that contain the specified text.

**Syntax:** [FileSearchEx](#) [dir](#), [file\\_mask](#), [find\\_string](#), [search\\_subdir](#), [return](#)

| Argument                      | Description   |
|-------------------------------|---|
| <a href="#">Dir</a>           | A string whose value is the full name of the directory where you want to search for files   |
| <a href="#">File_mask</a>     | A string whose value is the file mask describing existing files that you want to search. <a href="#">File_mask</a> can contain wildcard characters (* and ?).   |
| <a href="#">Find_string</a>   | A string whose value is the text contained in file(s) to be found   |
| <a href="#">Search_subdir</a> | A boolean that indicates whether you want to search in the subdirectories starting with <a href="#">dir</a> . Specify TRUE to search in the subdirectories, or FALSE to search only in the directory specified by <a href="#">dir</a> . |
| <a href="#">Return</a>        | A string variable that receives the returned value.   |

**Return value:** String. Returns comma-separated list of files that contain the specified [find\\_string](#).

**Usage:**

[FileSearchEx](#) statement should be used for searching in text files only, such as .TXT, .HTM, .LOG, and so on. It may produce incorrect results when searching in binary files.

You may want to use [FileSearchEx](#) to quickly scan various log files for common error messages.

**See also:**

- [FileReplaceEx](#)
- [FileFindFirst](#)
- [Pos](#)
- [InStr](#)

---

## [FileReplaceEx](#)

**Description:** Searches and replaces in files.

**Syntax:** [FileReplaceEx](#) [dir](#), [file\\_mask](#), [find\\_string](#), [replace\\_string](#), [search\\_subdir](#), [return](#)

| Argument                       | Description  |
|--------------------------------|--|
| <a href="#">Dir</a>            | A string whose value is the full name of the directory where you want to search for files  |
| <a href="#">File_mask</a>      | A string whose value is the file mask describing existing files that you want to search and replace. <a href="#">File_mask</a> can contain wildcard characters (* and ?).                  |
| <a href="#">Find_string</a>    | A string whose value is the text contained in file(s) to be found  |
| <a href="#">Replace_string</a> | A string whose value is the text that is used to replace all occurrences of the <a href="#">Find_string</a>  |
| <a href="#">Search_subdir</a>  | A boolean that indicates whether you want to search and replace in the subdirectories starting with <a href="#">dir</a> . Specify TRUE to search in the subdirectories, or FALSE to search |

|                        |  |
|------------------------|--|
|                        | only in the directory specified by <a href="#">dir</a> . |
| <a href="#">Return</a> | A numeric variable that receives the returned value.     |

**Return value:** Number. Returns total number of replacements made in all files that were found to contain the specified [find\\_string](#).

**Usage:** [FileReplaceEx](#) statement should be used for searching and replacing in text files only, such as .TXT, .HTM, .LOG, and so on. It may corrupt binary files.



**Tip:**

[FileReplaceEx](#) statement can be used to quickly convert files from UNIX format to DOS/Windows format and vice versa.

Note that UNIX files use NL character (ASCII code 10) as a end of line markers where DOS files use CR/NL pair of characters (ASCII codes 13 and 10) for this purpose. Use [FileReplaceEx](#) statement to convert files after FTP downloading from UNIX or before FTP uploading to UNIX hosts.

Examples:

UNIX to DOS: [FileReplaceEx](#) "\*.htm", "\n", "\r\n", False, count

DOS to UNIX: [FileReplaceEx](#) "\*.htm", "\r\n", "\n", False, count

**See also:**

[FileSearchEx](#)  
[FileFindFirst](#)  
[FileConvert](#)  
[Replace](#)

---

## FileConvert

**Description:** Converts the specified file by replacing some symbols.

**Syntax:** [FileConvert](#) file, [find\\_table](#), [replace\\_table](#), [return](#)

| Argument                       | Description  |
|--------------------------------|--|
| <a href="#">File</a>           | A string whose value is the full name of the file in which you want to replace symbols from <a href="#">Source_list</a> string with matching symbols from the <a href="#">Transform_list</a> string. |
| <a href="#">Source_list</a>    | A comma-separated string containing ASCII codes of symbols to be converted (replaced)  |
| <a href="#">Transform_list</a> | A comma-separated string containing ASCII codes of symbols to be used for conversion.  |
| <a href="#">Return</a>         | A numeric variable that receives the returned value.   |

**Return value:** Number. Returns number of replacements made in the [file](#).

**Usage:** Number of symbols in the [Transform\\_list](#) string must match number of symbols in the [Source\\_list](#) string. The [FileConvert](#) statement scans the specified [file](#) for symbols included into [Source\\_list](#). For every found occurrence, the [FileConvert](#) statement replaces the found symbol with the corresponding symbol from the [Transform\\_list](#). This allows you to perform multi-symbol "replace all" operation on a file.

[FileConvert](#) statement can process both text and binary files.

**Tip:**

You may use [FileConvert](#) statement to convert mainframe files using EBCDIC code table to files using ASCII code table. Translations between code sets can be tricky. There are a variety of problems that may occur. A problem related to translation is record format. In mainframe world, records exist as fixed-length, variable-length and string. There are no delimiters. All file types may contain text or binary data. When sent via FTP, however, there are no fixed-length records. Also, EBCDIC records are always ended by a new line (NL) character (X'15') and ASCII records are ended with a carriage return, line feed pair (CR LF). These characters cannot be embedded in the records. Binary files are treated as a single string of bytes and such files have no record structure. Thus, choice of translation affects the record format of the file. [FileConvert](#) statement provides powerful and yet simple options for translating between ASCII and EBCDIC. 24x7 Scheduler is shipped with the Example Jobs database that includes two user-defined JAL statements for performing EBCDIC to ASCII and ASCII to EBCDIC file conversions using common translation tables presented as translation lists.

You can customize translations lists in these example statements to make them suitable for your unique requirements.

**Examples:**

This statement replaces ASCII symbols 0 and 7 with a space (ASCII code 32) and a tab (ASCII code 8) symbols

```
FileConvert "c:\ftp\myfile.txt", "0,7", "32,8", count
```

This statement replaces ASCII symbol 0 with a space (ASCII code 32)

```
FileConvert "c:\ftp\myfile.txt", "0", "32", count
```

**Common Translation Tables****EBCDIC to ASCII**

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | a  | b  | c  | d  | e  | f  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 00 | 01 | 02 | 03 | DC | 09 | C3 | 7F | CA | B2 | D5 | 0B | 0C | 0D | 0E | 0F |
| 1 | 10 | 11 | 12 | 13 | DB | DA | 08 | C1 | 18 | 19 | C8 | F2 | 1C | 1D | 1E | 1F |
| 2 | C4 | B3 | C0 | D9 | BF | 0A | 17 | 1B | B4 | C2 | C5 | B0 | B1 | 05 | 06 | 07 |
| 3 | CD | BA | 16 | BC | BB | C9 | CC | 04 | B9 | CB | CE | DF | 14 | 15 | FE | 1A |
| 4 | 20 | FF | 83 | 84 | 85 | A0 | C6 | 86 | 87 | A4 | BD | 2E | 3C | 28 | 2B | 7C |
| 5 | 26 | 82 | 88 | 89 | 8A | A1 | 8C | 8B | 8D | E1 | 21 | 24 | 2A | 29 | 3B | AA |
| 6 | 2D | 2F | B6 | 8E | B7 | B5 | C7 | 8F | 80 | A5 | DD | 2C | 25 | 5F | 3E | 3F |
| 7 | 9B | 90 | D2 | D3 | D4 | D6 | D7 | D8 | DE | 60 | 3A | 23 | 40 | 27 | 3D | 22 |
| 8 | 9D | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | AE | AF | D0 | EC | E7 | F1 |
| 9 | F8 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | A6 | A7 | 91 | F7 | 92 | CF |
| a | E6 | 7E | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | AD | A8 | D1 | ED | E8 | A9 |
| b | 5E | 9C | BE | FA | B8 | F5 | F4 | AC | AB | F3 | 5B | 5D | EE | F9 | EF | 9E |
| c | 7B | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | F0 | 93 | 94 | 95 | A2 | E4 |
| d | 7D | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | FB | 96 | 81 | 97 | A3 | 98 |
| e | 5C | F6 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | FD | E2 | 99 | E3 | E0 | E5 |
| f | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | FC | EA | 9A | EB | E9 | 9F |

**ASCII to EBCDIC**

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | a  | b  | c  | d  | e  | f  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 00 | 01 | 02 | 03 | 37 | 2D | 2E | 2F | 16 | 05 | 25 | 0B | 0C | 0D | 0E | 0F |
| 1 | 10 | 11 | 12 | 13 | 3C | 3D | 32 | 26 | 18 | 19 | 3F | 27 | 1C | 1D | 1E | 1F |
| 2 | 40 | 5A | 7F | 7B | 5B | 6C | 50 | 7D | 4D | 5D | 5C | 4E | 6B | 60 | 4B | 61 |
| 3 | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | 7A | 5E | 4C | 7E | 6E | 6F |
| 4 | 7C | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | D1 | D2 | D3 | D4 | D5 | D6 |
| 5 | D7 | D8 | D9 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | BA | E0 | BB | B0 | 6D |
| 6 | 79 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 91 | 92 | 93 | 94 | 95 | 96 |
| 7 | 97 | 98 | 99 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | C0 | 4F | D0 | A1 | 07 |
| 8 | 68 | DC | 51 | 42 | 43 | 44 | 47 | 48 | 52 | 53 | 54 | 57 | 56 | 58 | 63 | 67 |
| 9 | 71 | 9C | 9E | CB | CC | CD | DB | DD | DF | EC | FC | 70 | B1 | 80 | BF | FF |
| a | 45 | 55 | CE | DE | 49 | 69 | 9A | 9B | AB | AF | 5F | B8 | B7 | AA | 8A | 8B |
| b | 2B | 2C | 09 | 21 | 28 | 65 | 62 | 64 | B4 | 38 | 31 | 34 | 33 | 4A | B2 | 24 |
| c | 22 | 17 | 29 | 06 | 20 | 2A | 46 | 66 | 1A | 35 | 08 | 39 | 36 | 30 | 3A | 9F |

```
d 8C AC 72 73 74 0A 75 76 77 23 15 14 04 6A 78 3B
e EE 59 EB ED CF EF A0 8E AE FE FB FD 8D AD BC BE
f CA 8F 1B B9 B6 B5 E1 9D 90 BD B3 DA FA EA 3E 41
```

**See also:**

- FileSearchEx
- FileReplaceEx
- Replace
- FTP statements
- Telnet statements

---

## FileGetAttr

**Description:** Reports file attributes for the specified file or directory.

**Syntax:** [FileGetAttr](#) file, return

| Argument               | Description  |
|------------------------|--|
| <a href="#">file</a>   | A string variable that is the file or directory name you want to query |
| <a href="#">return</a> | A numeric variable that receives the returned value                    |

**Return value:** Number. Returns a number representing the attributes of a file, directory, or folder.

**Usage:** The value returned by [FileGetAttr](#) is the sum of any following attribute values:

| Value | Description                        |
|-------|------------------------------------|
| 0     | Normal                             |
| 1     | Read-only                          |
| 2     | Hidden                             |
| 4     | System                             |
| 16    | Directory or folder                |
| 32    | File has changed since last backup |

To determine which attributes are set, use the BitwiseAnd statement to perform a bitwise comparison of the value returned by the [FileGetAttr](#) statement and the value of the individual file attribute you want. If the result is not zero, that attribute is set for the named file.

**See also:**

- FileSetAttr
- Bitwise statements

---

## FileSetAttr

**Description:** Sets file attributes for the specified file or directory.

**Syntax:** [FileSetAttr](#) file, attr

| Argument             | Description  |
|----------------------|--|
| <a href="#">file</a> | A string variable that is the file or directory name you want to query |
| <a href="#">attr</a> | A numeric variable that is the new file attributes                     |

**Return value:** None

**Usage:** The [attr](#) value for the [FileSetAttr](#) is the sum of any following attribute values:

| Value | Description                        |
|-------|------------------------------------|
| 0     | Normal                             |
| 1     | Read-only                          |
| 2     | Hidden                             |
| 4     | System                             |
| 16    | Directory or folder                |
| 32    | File has changed since last backup |



**Note:**

A run-time error occurs if you try to set the attributes of an open file.

**See also:**

[FileGetAttr](#)

---

## FileGetPos

**Description:** Reports current position in the specified file previously opened by [FileOpen](#) statement.

**Syntax:** [FileGetPos](#) filenum, pos

| Argument                | Description  |
|-------------------------|--|
| <a href="#">Filenum</a> | The file number previously assigned to the file when it was opened by <a href="#">FileOpen</a> statement |
| <a href="#">Pos</a>     | A numeric variable that receives the returned value  |

**Return value:** Number. Returns the file position after the read/write operation or zero if no operation has been performed after file opening.



**Note:**

[FileGetPos](#) always returns position from the beginning of the file.

**See also:**

[FileSetPos](#)

[FileOpen](#)

## FileSetPos

**Description:** Moves the file pointer to the specified position in a file previously opened by [FileOpen](#) statement. The file pointer is the position in the file at which the next read or write begins.

**Syntax:** [FileSetPos](#) [filename](#), [origin](#), [pos](#)

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">Filename</a> | The file number previously assigned to the file when it was opened by <a href="#">FileOpen</a> statement  |
| <a href="#">Pos</a>      | A number whose value is the new position of the file pointer relative to the position specified in <a href="#">origin</a> , in bytes  |
| <a href="#">origin</a>   | A string constant whose value specifies from where you want to set then position. Values are: <ul style="list-style-type: none"><li>• <a href="#">"START"</a> — At the beginning of the file</li><li>• <a href="#">"CURRENT"</a> — At the current position</li><li>• <a href="#">"END"</a> — At the end of the file</li></ul> |

**Return value:** None.

**Usage:** The file must be opened previously using [FileOpen](#) statement.

**See also:**

[FileGetPos](#)  
[FileOpen](#)

---

## FileOpen

**Description:** Opens the specified file for reading or writing and assigns it a unique file number. You use this number to identify the file when you read, write, or close the file. The arguments [filemode](#), [fileaccess](#), [filelock](#), and [append](#) determine the mode in which the file is opened. If the file doesn't exist, a new file is created.

**Syntax:** [FileOpen](#) [filename](#), [filemode](#), [fileaccess](#), [append](#) ,[return](#)

| Argument                   | Description  |
|----------------------------|--|
| <a href="#">filename</a>   | A string whose value is the name of the file you want to open. If <a href="#">filename</a> is not on the operating system's search path, you must enter the fully qualified name   |
| <a href="#">filemode</a>   | A string constant whose value specifies how the end of a file read or file write operation is determined. Values are: <ul style="list-style-type: none"><li>• <a href="#">"LineMode"</a> — Read or write the file a line at a time</li><li>• <a href="#">"StreamMode"</a> — Read the file in 32K chunks. For more information, see Usage below</li></ul> |
| <a href="#">fileaccess</a> | A string constant whose value specifies whether the file is opened for reading or writing. Values are: <ul style="list-style-type: none"><li>• <a href="#">"Read"</a> — Read-only access</li></ul>   |

|               |   |
|---------------|---|
|               | <ul style="list-style-type: none"><li>• <b>"Write"</b> — Write-only access</li></ul>  |
| <b>append</b> | <p>A boolean whose value specifies whether existing data in the file is overwritten when file is opened for write operation. Values are:</p> <ul style="list-style-type: none"><li>• True — Write data to the end of the file</li><li>• False — Replace all existing data in the file</li></ul> <p><b>append</b> is ignored if the <b>fileaccess</b> argument is "Read"</p> |
| <b>return</b> | <p>A numeric variable that receives the returned file number.</p>   |

**Return value:** Number. Returns the file number assigned to **filename**.

**Usage:** When a file has been opened in **"LineMode"**, each call to the **FileRead** statement reads until it encounters a carriage return (CR), linefeed (LF), or end-of-file mark (EOF). Each call to the **FileWrite** adds a CR and LF at the end of each string it writes.

When a file has been opened in **"StreamMode"**, a call to **FileRead** reads the whole file (until it encounters an EOF) or 32,765 bytes, whichever is less. **FileWrite** writes a maximum of 32,765 bytes in a single call and does not add CR and LF characters.



**Note:**

If the 24x7 Scheduler doesn't find the file, it creates a new file, giving it the specified name.

**See also:**

FileClose  
FileRead  
FileWrite

---

## FilePrint

**Description:** Sends the specified file to the default printer

**Syntax:** **FilePrint** **filename**

| Argument        | Description  |
|-----------------|--|
| <b>filename</b> | A string whose value is the name of the file you want to print. If <b>filename</b> is not on the operating system's search path, you must enter the fully qualified name |

**Return value:** None.

**See also:**

PrinterSetDefault

## FileRead

**Description:** Reads data from the file associated with the specified file number, which was assigned to the file with the [FileOpen](#) statement.

**Syntax:** [FileRead](#) *filenum*, *return*

| Argument                | Description  |
|-------------------------|--|
| <a href="#">filenum</a> | The file number previously assigned to the file when it was opened by <a href="#">FileOpen</a> statement |
| <a href="#">return</a>  | A string variable into which you want to read the data   |

**Return value:** String.

If the file is opened in Line mode, [FileRead](#) reads a line of the file (that is, until it encounters a CR, LF, or EOF). It stores the contents of the line in the specified variable, skips the line-end characters, and positions the file pointer at the beginning of the next line.

If the file was opened in Stream mode, [FileRead](#) reads to the end of the file or the next 32,765 bytes, whichever is shorter. [FileRead](#) begins reading at the file pointer, which is positioned at the beginning of the file when the file is opened for reading. If the file is longer than 32,765 bytes, [FileRead](#) automatically positions the pointer after each read operation so that it is ready to read the next chunk of data.

[FileRead](#) can read a maximum of 32,765 characters at a time. Therefore, before calling the [FileRead](#) function, call the [FileSize](#) statement to check the file size. If your system has file sharing or security restrictions, you may need to call [FileSize](#) before you call [FileOpen](#).

An end-of-file mark is a NULL character (ASCII value 0). Therefore, if the file being read contains null characters, [FileRead](#) will stop reading at the first null character, interpreting it as the end of the file.

**Usage:** The file must be opened previously using [FileOpen](#) statement.

**See also:**

[FileSize](#)

[FileOpen](#)

---

## FileReadAll

**Description:** Loads data from the specified file.

**Syntax:** [FileReadAll](#) *file*, *return*

| Argument               | Description   |
|------------------------|---|
| <a href="#">file</a>   | A string whose value is the name of the existing file that you want to read |
| <a href="#">return</a> | A string variable into which you want to read the data                      |

**Return value:** String. Loads entire file specified by [file](#) into a string variable specified by [return](#).

**See also:**

FileRead  
FileSave  
FileCopy

---

## FileReadLine

**Description:** Reads specified line from a ASCII file.

**Syntax:** `FileReadLine file, line, return`

| Argument            | Description   |
|---------------------|---|
| <code>file</code>   | A string whose value is the name of the existing file that you want to read                 |
| <code>line</code>   | A number whose value is the number of the line that want to read from the <code>file</code> |
| <code>return</code> | A string variable into which you want to read the data                                      |

**Return value:** String. Reads specified `line` from `file`.

**Usage:** `FileReadLine` opens `file` for reading in the Line Mode, reads the file until it reaches the specified `line`, then closes the file. This statement makes sense only for ASCII (text) files.

**See also:**

FileRead  
FileReadAll

---

## FileRename

**Description:** Renames an existing file.

**Syntax:** `FileRename oldname, newname`

| Argument             | Description  |
|----------------------|--|
| <code>oldname</code> | A string whose value is the name of existing file or directory that want to rename |
| <code>newname</code> | A string variable whose value is the new file name                                 |

**Return value:** None

**Usage:** If the [newname](#) file already exists, the statement fails. You can use [FileMove](#) to override an existing file. Both [oldname](#) and [newname](#) can include path to the file.

**See also:**

[FileCopy](#)

[FileMove](#)

---

## FileSplitName

**Description:** Splits up a full file name into two components consisting of path, file name with extension

**Syntax:** [FileSplitName](#) [fullname](#), [filepath](#), [filename](#)

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">fullname</a> | A string whose value is the full file name that may include file path and file extension          |
| <a href="#">filepath</a> | A string variable whose value is the returned file path including drive letter and directory name |
| <a href="#">filename</a> | A string variable whose value is the returned file name including file extension                  |

**Return value:** String and String

**Usage:** The specified file does not have to exist.

**See also:**

[FileCopy](#)

[FileMove](#)

---

## FileSave

**Description:** Saves data in the specified file.

**Syntax:** [FileSave](#) [file](#), [text](#)

| Argument             | Description  |
|----------------------|--|
| <a href="#">file</a> | A string variable whose value is the name of the file into which you want to save the <a href="#">text</a> |
| <a href="#">text</a> | A string whose value is the data want to save in the <a href="#">file</a>                                  |

**Return value:** None.

**Usage:** If the [file](#) already exists then [FileSave](#) overwrites it.

**See also:**

[FileRead](#)

[FileExists](#)

[FileCopy](#)

---

## FileSize

**Description:** Reports the length of a file in bytes.

**Syntax:** [FileSize](#) [file](#), [return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">file</a>   | A string whose value is the name of the file for which you want to know the length. If filename is not on the current application library search path, you must specify the fully qualified name |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. Returns the length in bytes of the file identified by [file](#).

---

## FileWrite

**Description:** Write data to the file associated with the specified file number, which was assigned to the file with the [FileOpen](#) statement.

**Syntax:** [FileWrite](#) [filenum](#), [s](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">filenum</a> | The file number previously assigned to the file when it was opened by <a href="#">FileOpen</a> statement |
| <a href="#">s</a>       | A string variable, which you want to save to the file associated with <a href="#">filenum</a>            |

**Return value:** None.

**Usage:**

[FileWrite](#) writes its data at the position identified by the file pointer. If the file was opened by [FileOpen](#) with the [writemode](#) argument set to "Replace", the file pointer is initially at the beginning of the file. After each call to [FileWrite](#), the pointer is immediately after the last write. If the file was opened with the [writemode](#) argument set to "Append", the file pointer is initially at the end of the file and moves to the end of the file after each write.

**FileWrite** sets the file pointer following the last character written. If the file was opened in "LineMode" , **FileWrite** writes a carriage return (CR) and linefeed (LF) after the last character in variable and places the file pointer after the CR and LF.

**FileWrite** can write only 32,766 bytes at a time, which includes the string terminator character. If the length of variable **s** exceeds 32,765, **FileWrite** writes the first 32,765 characters only.

**Usage:** The file must be opened previously using **FileOpen** statement.

**See also:**

FileRead

FileOpen

---

## FileZip

**Description:** Compresses existing files into a new ZIP file. Produced ZIP files are compatible with PKZIP, Winzip, InfoZip, and most other zipping utilities. PKZIP utility is not required for zipping operations.

**Syntax:** **FileZip** *destination*, *source*

| Argument           | Description  |
|--------------------|--|
| <i>destination</i> | A string whose value is the name of a ZIP file into which you want to archive the <i>source</i> file |
| <i>source</i>      | A string whose value is the comma-separated list of existing files you want to zip.                  |

**Return value:** None

**Usage:** If the *destination* file already exists, **FileZip** overwrites the existing file.



**Important Note:** Source file names may not include wildcard characters. If you need to find files using wildcards, call the **Dir** statement to build the list of files then pass it as the *source* to the **FileZip** statement.

**See also:**

FileUnzip

FileDeleteEx

Dir

CD

## FileUnzip

**Description:** Decompresses files from an existing ZIP file. (PKUNZIP utility is not required for unzipping operations)

**Syntax:** `FileUnzip source, destination`

| Argument                 | Description   |
|--------------------------|---|
| <code>source</code>      | A string whose value is the name of existing ZIP file.  |
| <code>destination</code> | A string whose value is the name of existing directory to which all archived files will be extracted from the <code>source</code> file. |

**Return value:** None

**Usage:** If the `destination` directory does not exist, the statement fails. Extracted files may overwrite existing files with the same names in the `destination` directory.

**See also:**

- FileZip
- Dir
- FileDeleteEX

---

## IniFileGetKey

**Description:** Obtains the string value of a setting in the profile file or system registry. A profile file usually has `.INI` extension.

**Syntax:** `IniFileGetKey inifile, section, key, return`

| Argument             | Description   |
|----------------------|---|
| <code>inifile</code> | A string whose value is the name of the profile file. If you do not specify a full path, <code>IniFileGetKey</code> uses the operating system's standard file search order to find the file                               |
| <code>section</code> | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in <code>section</code> . Section is not case-sensitive |
| <code>key</code>     | A string specifying the setting name in section whose value you want. The setting name is followed by an equal sign in the file. Do not include the equal sign in <code>key</code> . Key is not case-sensitive            |
| <code>return</code>  | A string variable, which receives the returned value  |

**Return value:** String, with a maximum length of 4096 characters. Returns the string from [key](#) within [section](#) within [inifile](#). If [inifile](#) is not found, [section](#) is not found in [inifile](#), or [key](#) is not found in [section](#), [IniFileGetKey](#) returns the empty string ("" ). If an error occurs, it returns the empty string ("" ).

**Usage:** Use [IniFileGetKey](#) to get configuration settings from a profile file. You can use [IniFileSetKey](#) statement to change configuration of the application that owns this profile file. Before you make changes, you can use [IniFileGetKey](#) to obtain the original settings so you can restore them after or during the application run.

[IniFileGetKey](#) can also be used to obtain configuration settings from the Windows system registry. To obtain information from the registry instead of from an initialization file:

- On Windows NT, create a new folder called INIFILEMAPPING at the following location:  
[hkey\\_current\\_user\software\microsoft\windows nt\current version](#) To override the WIN.INI file, create a key in the new folder called WIN.INI with the following value: [#usr:software\microsoft\windows nt\current version\extensions](#)
- On Windows 95/98, substitute [windows](#) for [windows nt](#) in both paths (see above).

This change will tell Windows to override initialization files, so that you can use these initialization files to obtain registry settings.

**See also:**

[IniFileSetKey](#)  
[RegistryGetKey](#)  
[RegistrySetKey](#)

---

## IniFileSetKey

**Description:** Changes the string value of a setting in the profile file or system registry. A profile file usually has [.INI](#) extension.

**Syntax:** [IniFileSetKey](#) [inifile](#), [section](#), [key](#), [newvalue](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">inifile</a>  | A string whose value is the name of the profile file. If you do not specify a full path, <a href="#">IniFileSetKey</a> uses the operating system's standard file search order to find the file                               |
| <a href="#">section</a>  | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in <a href="#">section</a> . Section is not case-sensitive |
| <a href="#">key</a>      | A string specifying the setting name in section whose value you want to change. The setting name is followed by an equal sign in the file. Do not include the equal sign in <a href="#">key</a> . Key is not case-sensitive  |
| <a href="#">newvalue</a> | A string whose value is the value you want to specify for <a href="#">key</a> . The value can be up to 4096 characters long.   |

**Return value:** None.

**Usage:** [IniFileSetKey](#) changes the value for [key](#) within [section](#) within [inifile](#). If [inifile](#) is not found, [section](#) is not found in [inifile](#), or [key](#) is not found in [section](#), an error occurs.

Use [IniFileSetKey](#) to change configuration settings stored within profile file. You can use [IniFileSetKey](#) statement to change configuration of the application that owns this profile file. Before you make changes, you can use [IniFileGetKey](#) to obtain the original settings so you can restore them after or during the application run.

[IniFileSetKey](#) can also be used to change configuration settings in the Windows system registry. To change information in the registry instead of changing an initialization file:

- On Windows NT, create a new folder called INIFILEMAPPING at the following location:  
[hkey\\_current\\_user\software\microsoft\windows nt\current version](#) To override the WIN.INI file, create a key in the new folder called WIN.INI with the following value: [#usr:software\microsoft\windows nt\current version\extensions](#)
- On Windows 95/98, substitute [windows](#) for [windows nt](#) in both paths (see above).

This change will tell Windows to override initialization files, so that you can use these initialization files to obtain registry settings.

**See also:**

[IniFileGetKey](#)  
[RegistryGetKey](#)  
[RegistrySetKey](#)

---

## NotFileExists

**Description:** Reports whether the specified file does not exist.

**Syntax:** [NotFileExists](#) file, return

| Argument               | Description   |
|------------------------|---|
| <a href="#">file</a>   | A string whose value is the name of a file          |
| <a href="#">return</a> | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the file does not exist, FALSE if it exists.

**Usage:** If [file](#) is locked by another application, causing a sharing violation, [NotFileExists](#) also returns TRUE. [File](#) name may include wildcard characters (\* and ?).

**See also:**

[FileExists](#)

---

## RemoteDir

**Description:** Returns comma-separated list of files in the specified directory on the specified remote computer running 24x7 Remote Agent

**Syntax:** [RemoteDir](#) agent, file\_mask, return

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Agent</a>     | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile.   |
| <a href="#">File_mask</a> | A string whose value is the file mask that you want to use to list file. <a href="#">File_mask</a> can contain wildcard characters (* and ?). <a href="#">File_mask</a> can contain full or partial file path. |
| <a href="#">Return</a>    | A string variable that receives the returned value.  |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The [RemoteDir](#) statement is equivalent to DOS *dir* command executed on the remote computer hosting 24x7 Remote Agent specified by [Agent](#).

If you don't include file path to the [file\\_mask](#) then [RemoteDir](#) statement returns files from the [Agent](#)'s computer current directory.

**See also:**

FileTransfer  
Dir  
FTPDir

---

## File replication and synchronization statements

---

### SyncFTPDir

**Description:** Synchronizes and replicates files across two directories residing on local and remote computers using FTP protocol.

**Syntax:** [SyncFTPDir](#) [master](#), [server](#), [user](#), [password](#), [source\\_dir](#), [target\\_dir](#), [add\\_new](#), [delete\\_missing](#), [update\\_old](#), [subdir](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">Master</a> | <p>A string whose value instructs 24x7 Schedule which computer is the "master" computer containing files and subdirectories that you want to replicate.</p> <p>The following values are supported:</p> <ul style="list-style-type: none"><li>• <a href="#">"REMOTE"</a> - the remote FTP <a href="#">server</a> is "master"</li><li>• <a href="#">"LOCAL"</a> the local computer is "master"</li></ul> |
| <a href="#">Server</a> | A string whose value is host name of an FTP server (for example, <a href="#">ftp.microsoft.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> )   |

|                |  |
|----------------|--|
| User           | A string whose value is the name of the user to log on   |
| Password       | A string whose value is the password to use to log on  |
| Source_dir     | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate  |
| Target_dir     | A string whose value is the name of the target directory to which .files and subdirectories are replicated   |
| Add_new        | A boolean whose value should be TRUE if you want to replicate files that exist in the <a href="#">source_dir</a> , and FALSE otherwise   |
| Delete_missing | A boolean whose value should be TRUE if you want to delete from the <a href="#">target_dir</a> directory these files that <a href="#">target_dir</a> but do not exist in the <a href="#">source_dir</a> , and FALSE otherwise  |
| Update_old     | A boolean whose value should be TRUE if you want to update older versions of files in the <a href="#">target_dir</a> directory, and FALSE otherwise. If file is considered as old if it exists in both <a href="#">target_dir</a> and <a href="#">source_dir</a> directories and <a href="#">target_dir</a> version of that files has a time older than the version from the <a href="#">source_dir</a> directory. |
| Subdir         | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the <a href="#">Source_dir</a> .   |

**Return value:** None.

**Usage:** Use [SyncFTPDir](#) statement to automate synchronization and replication for a group of files residing on different computers.

The directory you are copying files is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.



**Important Note:** Because FTP protocol lacks file date/time manipulations, files uploaded to a FTP server always have system date/time on the FTP server computer (date/time when they were uploaded).



**Tips:**

- To update only outdated files set [Update\\_old](#) to [TRUE](#) and set both [Add\\_new](#) and [Delete\\_missing](#) to [FALSE](#).
- To perform full 2 way file synchronization between directory <one> residing on the remote computer and directory <two> residing on the local computer:  
First run [SyncFTPDir](#) using "REMOTE" for the [Master](#) and [Delete\\_missing](#) set to [FALSE](#) and everything else set to [TRUE](#).  
After that repeat [SyncFTPDir](#) using "LOCAL" for the [Master](#) and [Delete\\_missing](#) set to [FALSE](#) and everything else set to [TRUE](#).

**See also:**

SyncRemoteDir  
SyncLocalDir  
Other FTP statements

## SyncLocalDir

**Description:** Synchronizes and replicates files across two local or shared network directories.

**Syntax:** `SyncLocalDir source_dir, target_dir, add_new, delete_missing, update_old, subdir`

| Argument                    | Description   |
|-----------------------------|---|
| <code>Source_dir</code>     | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate   |
| <code>Target_dir</code>     | A string whose value is the name of the target directory to which .files and subdirectories are replicated  |
| <code>Add_new</code>        | A boolean whose value should be TRUE if you want to replicate files that exist in the <code>source_dir</code> , and FALSE otherwise   |
| <code>Delete_missing</code> | A boolean whose value should be TRUE if you want to delete from the <code>target_dir</code> directory these files that <code>target_dir</code> but do not exist in the <code>source_dir</code> , and FALSE otherwise  |
| <code>Update_old</code>     | A boolean whose value should be TRUE if you want to update older versions of files in the <code>target_dir</code> directory, and FALSE otherwise. If file is considered as old if it exists in both <code>target_dir</code> and <code>source_dir</code> directories and <code>target_dir</code> version of that files has a time older than the version from the <code>source_dir</code> directory. |
| <code>Subdir</code>         | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the <code>Source_dir</code> .   |

**Return value:** None.

**Usage:** Use `SyncLocalDir` statement to automate synchronization and replication for a group of files. The directory you are copying files is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.



**Tip:**

- To update only outdated files set `Update_old` to `TRUE` and set both `Add_new` and `Delete_missing` to `FALSE`.
- To perform full 2 way file synchronization between directory <one> and another directory <two>:  
First run `SyncLocalDir` using <one> as the "master" with `Delete_missing` set to `FALSE` and everything else set to `TRUE`.  
After that repeat `SyncLocalDir` using <two> as the "master" with `Delete_missing` set to `FALSE` and everything else set to `TRUE`.

**See also:**

SyncFTPDir  
SyncRemoteDir  
FileCopyEx  
FileMoveEx

---

## SyncRemoteDir

**Description:** Synchronizes and replicates files across two directories residing on local and remote computers provided 24x7 Remote Agent is running at a given remote computer.

**Syntax:** `SyncRemoteDir master, agent, source_dir, target_dir, add_new, delete_missing, update_old, subdir`

| Argument       | Description   |
|----------------|---|
| Master         | A string whose value instructs 24x7 Schedule which computer is the "master" computer containing files and subdirectories that you want to replicate.<br><br>The following values are supported: <ul style="list-style-type: none"><li>• "REMOTE" - the remote computer is "master"</li><li>• "LOCAL" the local computer is "master"</li></ul>   |
| Agent          | A string whose value is the name of 24x7 Remote Agent as it appears in the Remote Agent profile   |
| Source_dir     | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate   |
| Target_dir     | A string whose value is the name of the target directory to which .files and subdirectories are replicated  |
| Add_new        | A boolean whose value should be TRUE if you want to replicate files that exist in the <code>source_dir</code> , and FALSE otherwise   |
| Delete_missing | A boolean whose value should be TRUE if you want to delete from the <code>target_dir</code> directory these files that <code>target_dir</code> but do not exist in the <code>source_dir</code> , and FALSE otherwise  |
| Update_old     | A boolean whose value should be TRUE if you want to update older versions of files in the <code>target_dir</code> directory, and FALSE otherwise. If file is considered as old if it exists in both <code>target_dir</code> and <code>source_dir</code> directories and <code>target_dir</code> version of that files has a time older than the version from the <code>source_dir</code> directory. |
| Subdir         | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the <code>Source_dir</code> .   |

**Return value:** None.

**Usage:** Use `SyncRemoteDir` statement to automate synchronization and replication for a group of files residing on different computers.

The directory you are copying files is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.

**Tip:**

- To update only outdated files set [Update\\_old](#) to [TRUE](#) and set both [Add\\_new](#) and [Delete\\_missing](#) to [FALSE](#).
- To perform full 2 way file synchronization between directory <one> residing on the remote computer and directory <two> residing on the local computer:  
First run [SyncRemoteDir](#) using "REMOTE" for the [Master](#) and [Delete\\_missing](#) set to [FALSE](#) and everything else set to [TRUE](#).  
After that repeat [SyncRemoteDir](#) using "LOCAL" for the [Master](#) and [Delete\\_missing](#) set to [FALSE](#) and everything else set to [TRUE](#).

**See also:**

[SyncFTPDDir](#)  
[SyncLocalDir](#)  
[FileTransfer](#)

---

## Dir

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** [Dir](#) [file\\_mask](#), [return](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">File_mask</a> | A string whose value is the DOS file mask that you want to use to list file. <a href="#">File_mask</a> can contain wildcard characters (* and ?). <a href="#">File_mask</a> can contain full or partial file path. |
| <a href="#">Return</a>    | A string variable that receives the returned value.  |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The [Dir](#) statement is equivalent to DOS *dir* command.

To get the list of all files use \*.\* file mask. If you don't include file path to the [file\\_mask](#) then [Dir](#) statement returns files from the DOS current directory.

**See also:**

[CD](#)  
[FileSearchEx](#)  
[FileFindFirst](#)  
[FTPDDir](#)  
[RemoteDir](#)

---

## FTPDDir

**Description:** Returns comma-separated list of files in the specified directory on the specified FTP server

**Syntax:** [FTPDDir](#) [server](#), [user](#), [password](#), [file\\_mask](#), [return](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Server</a>    | A string whose value is host name of an FTP server (for example, <a href="#">ftp.microsoft.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> )   |
| <a href="#">User</a>      | A string whose value is the name of the user to log on   |
| <a href="#">Password</a>  | A string whose value is the password to use to log on  |
| <a href="#">File_mask</a> | A string whose value is the file mask that you want to use to list file. <a href="#">File_mask</a> can contain wildcard characters (* and ?). <a href="#">File_mask</a> can contain full or partial file path.<br><br><a href="#">File_mask</a> must be a valid FTP host Operation System file mask. |
| <a href="#">Return</a>    | A string variable that receives the returned value.  |

**Return value:** String. Returns comma-separated list of files.

**Usage:** On DOS/Windows based FTP hosts the [FTPDDir](#) statement is equivalent to DOS *dir* command. For most UNIX flavors, the [FTPDDir](#) statement is equivalent to UNIX *ls* command.

If you don't include file path to the [file\\_mask](#) then [FTPDDir](#) statement returns files from the FTP server current directory.

**See also:**

- [FTPFileExists](#)
- [Dir](#)
- [RemoteDir](#)

---

## RemoteDir

**Description:** Returns comma-separated list of files in the specified directory on the specified remote computer running 24x7 Remote Agent

**Syntax:** [RemoteDir](#) [agent](#), [file\\_mask](#), [return](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Agent</a>     | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile.   |
| <a href="#">File_mask</a> | A string whose value is the file mask that you want to use to list file. <a href="#">File_mask</a> can contain wildcard characters (* and ?). <a href="#">File_mask</a> can contain full or partial file path. |
| <a href="#">Return</a>    | A string variable that receives the returned value.  |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The [RemoteDir](#) statement is equivalent to DOS *dir* command executed on the remote computer hosting 24x7 Remote Agent specified by [Agent](#).  
If you don't include file path to the [file\\_mask](#) then [RemoteDir](#) statement returns files from the [Agent](#)'s computer current directory.

**See also:**

FileTransfer  
Dir  
FTPDir

---

## FTP statements

---

### FTPConfig

**Description:** Set various parameters for the subsequent FTP operations.

**Syntax:** [FTPConfig](#) *property*, *new\_value*

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Property</a>  | A string whose value is the name of the property for the FTP session that you want to change. The following properties are supported: <ul style="list-style-type: none"><li>• <a href="#">"TRANSFER MODE"</a></li></ul>                          |
| <a href="#">New_value</a> | A string whose value is the new value for the <a href="#">property</a> that you want to change. The following values are supported: <ul style="list-style-type: none"><li>• <a href="#">"ASCII"</a></li><li>• <a href="#">"BINARY"</a></li></ul> |

**Return value:** None.

**Usage:** Use [FTPConfig](#) statement to configure transfer mode when downloading or uploading files using [FTPGetFile](#) and [FTPPutFile](#) statements. Both parameter name and new value are case insensitive.

Transfer type setting determines which transfer mode will be set when you initiate a file transfer. It can be one of the following:

**ASCII:** This mode is used for transferring text files between sites running different operating systems. It will take care of all the necessary translation to make text readable. However, use it with caution since binary files will be corrupted if transferred in this mode. Text files with non-English characters in it will also be corrupted.

**Binary:** This is the most frequently used transfer mode. It transfers raw data without any translation.

**See also:**

FTPGetFile

---

## FTPDDir

**Description:** Returns comma-separated list of files in the specified directory on the specified FTP server

**Syntax:** `FTPDDir server, user, password, file_mask, return`

| Argument  | Description   |
|-----------|---|
| Server    | A string whose value is host name of an FTP server (for example, <code>ftp.microsoft.com</code> ) or the IP number of the site in ASCII dotted-decimal format (for example, <code>11.0.1.45</code> )  |
| User      | A string whose value is the name of the user to log on  |
| Password  | A string whose value is the password to use to log on   |
| File_mask | A string whose value is the file mask that you want to use to list file. <code>File_mask</code> can contain wildcard characters (* and ?). <code>File_mask</code> can contain full or partial file path.<br><code>File_mask</code> must be a valid FTP host Operation System file mask. |
| Return    | A string variable that receives the returned value.   |

**Return value:** String. Returns comma-separated list of files.

**Usage:** On DOS/Windows based FTP hosts the `FTPDDir` statement is equivalent to DOS `dir` command. For most UNIX flavors, the `FTPDDir` statement is equivalent to UNIX `ls` command.

If you don't include file path to the `file_mask` then `FTPDDir` statement returns files from the FTP server current directory.

**See also:**

- FTPFileExists
- Dir
- RemoteDir

---

## FTPDeleteFile

**Description:** Deletes the specified file on the specified remote FTP server.

**Syntax:** `FTPDeleteFile server, user, password, file`

| Argument | Description   |
|----------|---|
| Server   | A string whose value is host name of an FTP server (for |

|          |  |
|----------|--|
|          | example, <a href="ftp.microsoft.com">ftp.microsoft.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| User     | A string whose value is the name of the user to log on   |
| password | A string whose value is the password to use to log on  |
| file     | A string whose value is the name of the file that you want to delete   |

**Return value:** None.

**Usage:** [file](#) can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The [FTPDeleteFile](#) statement translates the directory name separators to the appropriate character before they are used.



**Note:**

[FTPDeleteFile](#) statement uses the default port for FTP servers



**Tip:**

You can have [FTPDeleteFile](#) statement to delete multiple files in one pass. This is more efficient than calling [FTPDeleteFile](#) for every individual file as it requires a single FTP connection. To delete multiple files in one pass, specify the [file](#) names as a comma separated list.

**See also:**

FTPGetFile

---

## FTPFileExists

**Description:** Reports whether the specified file exists on the specified remote FTP server.

**Syntax:** [FTPFileExists](#) [server](#), [user](#), [password](#), [file](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">server</a>   | A string whose value is host name of an FTP server (for example, <a href="ftp.microsoft.com">ftp.microsoft.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| <a href="#">user</a>     | A string whose value is the name of the user to log on   |
| <a href="#">password</a> | A string whose value is the password to use to log on  |
| <a href="#">file</a>     | A string whose value is the name of the file that you want to check  |
| <a href="#">return</a>   | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns TRUE if the file exists, FALSE if it does not exist.

**Usage:** [file](#) can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The [FTPFileExists](#) statement translates the directory name separators to the appropriate character before they are used.



**Notes:**

- [FTPFileExists](#) statement uses the default port for FTP servers

- File name is case-sensitive on all FTP server platforms. For example, the result of file checking will be negative if search for a file "data.txt" and the actual file name reported by the FTP server is "DATA.TXT" or "Data.txt" .

**See also:**

FTPGetFile  
FTPDir

---

## FTPGetFile

**Description:** Retrieves a file from the specified FTP server and stores it under the specified file name, creating a new local file in the process.

**Syntax:** `FTPGetFile server, user, password, source, target`

| Argument              | Description  |
|-----------------------|--|
| <code>server</code>   | A string whose value is host name of an FTP server (for example, <code>ftp.microsoft.com</code> ) or the IP number of the site in ASCII dotted-decimal format (for example, <code>11.0.1.45</code> ) |
| <code>user</code>     | A string whose value is the name of the user to log on   |
| <code>password</code> | A string whose value is the password to use to log on  |
| <code>source</code>   | A string whose value is the name of the file to retrieve from the remote system  |
| <code>target</code>   | A string whose value is the name of the file to create on the local system   |

**Return value:** None.

**Usage:** Both `source` and `target` file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The `FTPGetFile` statement translates the directory name separators to the appropriate character before they are used. By default `FTPGetFile` statement uses binary transfer mode. To change transfer mode to ASCII, use `FTPConfig` statement.

**Note:**

`FTPGetFile` statement uses the default port for FTP servers

**Tip:**

You can have `FTPGetFile` statement to retrieve multiple files in one pass. This is more efficient then calling `FTPGetFile` for every individual file as it requires a single FTP connection. To retrieve multiple files in one pass, specify the `source` files as a comma separated list. The `target` files must be also specified as a comma separated list. Make sure to specify matching number of file names in the `source` and `target` file lists.

**See also:**

FTPPutFile  
FTPConfig  
SyncFTPDir

## FTPPutFile

**Description:** Transfers a file from the local system to specified remote FTP server and stores it under the specified file name, creating a new remote file in the process.

**Syntax:** `FTPPutFile server, user, password, source, target`

| Argument              | Description  |
|-----------------------|--|
| <code>server</code>   | A string whose value is host name of an FTP server (for example, <code>ftp.microsoft.com</code> ) or the IP number of the site in ASCII dotted-decimal format (for example, <code>11.0.1.45</code> ) |
| <code>user</code>     | A string whose value is the name of the user to log on   |
| <code>password</code> | A string whose value is the password to use to log on  |
| <code>source</code>   | A string whose value is the name of the file to transfer from the local system   |
| <code>target</code>   | A string whose value is the name of the file to create on the remote system  |

**Return value:** None.

**Usage:** Both `source` and `target` file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The `FTPPutFile` statement translates the directory name separators to the appropriate character before they are used. By default `FTPPutFile` statement uses binary transfer mode. To change transfer mode to ASCII, use `FTPConfig` statement.



**Note:**

`FTPPutFile` statement uses the default port for FTP servers



**Tip:**

You can have `FTPPutFile` statement to transfer multiple files in one pass. This is more efficient than calling `FTPPutFile` for every individual file as it requires a single FTP connection. To transfer multiple files in one pass, specify the `source` files as a comma separated list. The `target` files must be also specified as a comma separated list. Make sure to specify matching number of file names in the `source` and `target` file lists.

**See also:**

- FTPGetFile
- FTPConfig
- SyncFTPDir

---

## FTPRenameFile

**Description:** Renames the specified remote file on the specified FTP server.

**Syntax:** `FTPRenameFile server, user, password, oldname, newname`

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Server</a>   | A string whose value is host name of an FTP server (for example, <a href="#">ftp.microsoft.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| <a href="#">User</a>     | A string whose value is the name of the user to log on   |
| <a href="#">Password</a> | A string whose value is the password to use to log on  |
| <a href="#">Oldname</a>  | A string whose value is the name of the file to rename   |
| <a href="#">Newname</a>  | A string whose value is the new name of the file   |

**Return value:** None.

**Usage:** Both [oldname](#) and [newname](#) file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The [FTPRenameFile](#) statement translates the directory name separators to the appropriate character before they are used.



**Note:**

[FTPRenameFile](#) statement uses the default port for FTP servers

**See also:**

FTPPutFile  
FTPGetFile  
FTPDeleteFile

---

## Job management statements

---

### JobCreate

**Description:** Creates a new job.

**Syntax:** [JobCreate](#) [return](#)

| Argument               | Description |
|------------------------|-------------|
| <a href="#">Return</a> | Number      |

**Return value:** Number. Returns job ID for the new job. The new job inherits default job properties.

**Usage:** Use [JobCreate](#) in your scripts to programmatically create new jobs. Use [JobModify](#) statement to setup properties of the created job . Use [JobEnable](#) statement with the [TRUE](#) status to enable the job after you done with setting job properties.

**Important note:**

- The [JobCreate](#) statement is not supported in asynchronous and remote jobs. An error will occur if you use this statement.

**See also:**

[JobDelete](#)  
[JobEnable](#)  
[JobModify](#)

---

## JobDelete

**Description:** Deletes the specified job.

**Syntax:** [JobDelete](#) job

| Argument            | Description                                      |
|---------------------|--|
| <a href="#">Job</a> | A string whose value is ether job ID or job name |

**Return value:** None

**Usage:** Use [JobDelete](#) in your scripts to programmatically delete jobs. [JobDelete](#) permanently deletes the specified job from the active job pool. The deleted job cannot be recovered at a later time. Use [JobEnable](#) statement with the [FALSE](#) state to temporary disable jobs. If the specified [job](#) is not found, an error occurs.

**Important note:**

- The [JobDelete](#) statement is not supported in asynchronous and remote jobs. An error "Job not found" will occur if you use this statement.

**See also:**

[JobEnable](#)  
[JobGetStatus](#)

---

## JobDescribe

**Description:** Describes properties of the specified job.

**Syntax:** [JobDescribe](#) job, property, value

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">job</a>      | A string whose value is ether job ID or job name  |
| <a href="#">property</a> | A string whose value is the property that you want to modify. See Job Properties topic for list of available properties |

|       |        |
|-------|--------|
| value | Return |
|-------|--------|

**Return value:** A string whose value is the value for the [property](#). The returned value is always converted to the string.

**Usage:** Use this statement to retrieve job definitions, job schedules and triggers dynamically from the script. For example, you can use [JobDescribe](#) to create customizable job monitors.



**Notes:**

- The [JobDescribe](#) statement is not supported in asynchronous and remote jobs. An error “Job not found” will occur if you use this statement.

**See also:**

[JobModify](#)  
[JobEnable](#)  
[Job Properties](#)

---

## JobEnable

**Description:** Enables or disables the specified job.

**Syntax:** [JobEnable](#) job, state

| Argument              | Description  |
|-----------------------|--|
| <a href="#">job</a>   | A string whose value is ether job ID or job name   |
| <a href="#">state</a> | A boolean whose value is the new job state. Specify <a href="#">TRUE</a> to enable the <a href="#">job</a> or <a href="#">FALSE</a> to disable the <a href="#">job</a> |

**Return value:** None

**Usage:** Use this statement to enable/disable jobs in the active job pool. If the specified [job](#) is not found, an error occurs.



**Notes:**

- Jobs disabled by using [JobEnable](#) statement remain in the active job pool until job database “save” is invoked via 24x7 GUI or via external interface. On “save” all disabled jobs are purged from the active job pool. Jobs disabled via 24x7 GUI are purged from the active job pool right away. However, disabled jobs are not removed from the job database. To physically remove a job from the job database use 24x7 GUI or one of supported external interfaces.
- The [JobEnable](#) statement is not supported in asynchronous and remote jobs. An error “Job not found” will occur if you use this statement.

**See also:**

[JobRun](#)  
[JobModify](#)

## JobGetStatus

**Description:** Reports status for the specified job.

**Syntax:** `JobGetStatus job, return`

| Argument            | Description                                       |
|---------------------|---|
| <code>job</code>    | A string whose value is either job ID or job name |
| <code>return</code> | A numeric whose value is the job status.          |

**Return value:** Number. Returned job status can be one of the following:

|                  |   |
|------------------|---|
| -1               | Error job. Job may have this status because of incomplete job definition or if an error occurred while executing this job.  |
| -2               | Job is performing notification action after or before job run, such as sending email message, executing database command, etc.  |
| -3               | Job is running.   |
| -5               | First job run is pending. Such job was never started before.  |
| -6               | Unknown status. Usually job has an unknown status when it is asynchronous job of "Program" type after the job has started the specified program.  |
| 0                | Successfully finished. Note: an asynchronous job of "Program" type also may have this status after it successfully started the specified program but before the program actually stops running. |
| Any other number | Successfully finished (for asynchronous jobs of "Program" type only). A job may have this status after it successfully started the specified program and the program is still running.          |

**Usage:** Use this statement to retrieve job status programmatically from the script. This statement can be useful in building various job monitors.



**Notes:**

- Use `JobDescribe` statement to found out job Enabled/Disabled state.
- You cannot change job status directly. Job status is the internal job state maintained by the 24x7 Scheduler.
- The `JobGetStatus` statement is not supported in asynchronous and remote jobs. An error "Job not found" will occur if you use this statement.
- An asynchronous job of "Program" type may have 0 status even if the started "program" (process) is still running in any of the following cases: started program spawned one or more child processes and then main entry process terminated; started program launched other separate processes and then terminated; started program is not responding or has security attributes that do not allow querying status of the process.

**See also:**

JobRun  
JobDescribe

## JobModify

**Description:** Modifies properties of the specified job.

**Syntax:** `JobModify job, property, new_value`

| Argument               | Description   |
|------------------------|---|
| <code>job</code>       | A string whose value is either job ID or job name   |
| <code>property</code>  | A string whose value is the property that you want to modify. See Job Properties topic for list of available properties |
| <code>new_value</code> | A string whose value is the new value for the <code>property</code>   |

**Return value:** None

**Usage:** Use this statement to modify job definitions, job schedules and triggers dynamically from the script. For example, you can use `JobModify` to reschedule jobs for a later run when the exact start time is unknown in design-time.



**Notes:**

- Modified jobs are updated immediately in both places: the active job pool and the job database. 24x7 Scheduler internally triggers "save" command after every change.
- Side effect: On "save," all disabled jobs are purged from the active job pool. However, disabled jobs are not removed from the job database. To physically remove a job from the job database use 24x7 GUI or one of supported external interfaces.
- The `JobModify` statement is not supported in asynchronous and remote jobs. An error "Job not found" will occur if you use this statement.

**See also:**

`JobModify`  
`JobDescribe`  
`JobEnable`  
Job Properties

---

## JobRun

**Description:** Runs the specified job.

**Syntax:** `JobRun job`

| Argument         | Description                                       |
|------------------|---|
| <code>job</code> | A string whose value is either job ID or job name |

**Return value:** None

**Usage:** Use [JobRun](#) in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs.

[JobRun](#) can start jobs from the active job pool only. If the specified [job](#) is not found, an error occurs.



**Important note:**

- The [JobRun](#) statement is not supported in asynchronous and remote jobs. An error “Job not found” will occur if you use this statement.

**See also:**

[JobEnable](#)  
[JobGetStatus](#)  
[JobRemoteRun](#)

---

## JobRemoteRun

**Description:** Runs the specified job using the specified remote agent.

**Syntax:** [JobRemoteRun](#) [job](#), [agent](#)

| Argument              | Description   |
|-----------------------|---|
| <a href="#">Job</a>   | A string whose value is ether job ID or job name  |
| <a href="#">Agent</a> | A string whose value is the name of 24x7 Remote Agent as it appears in the Remote Agent profile |

**Return value:** None

**Usage:**

Use [JobRemoteRun](#) in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs.

[JobRemoteRun](#) allows dynamic selection for the Remote Agent that will execute the specified [job](#). [JobRemoteRun](#) ignores name of the default Remote Agent that may be assigned to the [job](#) in the job properties, where [JobRun](#) statement instead uses the default Remote Agent if there is any.

[JobRemoteRun](#) can start jobs from the active job pool only. If the specified [job](#) is not found, an error occurs.



**Important note:**

- The [JobRemoteRun](#) statement is not supported in asynchronous and remote jobs. An error “Job not found” will occur if you use this statement.

**See also:**

[JobEnable](#)  
[JobGetStatus](#)  
[JobRun](#)  
[GetRemoteVariable](#)  
[SetRemoteVariable](#)

## GetRemoteVariable

**Description:** Obtains value of the specified global variable on the specified 24x7 Remote Agent or 24x7 Master Scheduler.

**Syntax:** `GetRemoteVariable agent, variable, return`

---

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Agent</a>    | A string whose value is the name of 24x7 Remote Agent as it appears in the Remote Agent profile                                    |
| <a href="#">Variable</a> | A string whose value is the name of the global variable that must exist in the run-time environment of the <a href="#">Agent</a> . |
| <a href="#">Return</a>   | A string variable that receives the returned value.  |

---

**Return value:** Returns value of a global variable stored in the run-time environment of the [Agent](#). The value is returned in the string format regardless of the data type of the remote [variable](#).

**Usage:**

Use [GetRemoteVariable](#) and [SetRemoteVariable](#) statements to pass data between networked 24x7 Scheduler components. This can be used to build complex distributed jobs. This can be also used for synchronization of jobs running simultaneously on different computers.

---

**See also:**

- [Dim](#)
- [Set](#)
- [SetRemoteVariable](#)
- [JobGetStatus](#)
- [JobRemoteRun](#)

---

## SetRemoteVariable

**Description:** Sets value of the specified global variable on the specified 24x7 Remote Agent or 24x7 Master Scheduler.

**Syntax:** `SetRemoteVariable agent, variable, new_value`

---

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Agent</a>     | A string whose value is the name of 24x7 Remote Agent as it appears in the Remote Agent profile  |
| <a href="#">Variable</a>  | A string whose value is the name of the global variable that must exist in the run-time environment of the <a href="#">Agent</a> .   |
| <a href="#">New_Value</a> | A constant or a variable whose value you want to store in the remote <a href="#">variable</a> . The data type of the <a href="#">new_value</a> must match the data type of the remote global variable. |

---

**Return value:** None.

**Usage:**

Use [GetRemoteVariable](#) and [SetRemoteVariable](#) statements to pass data between networked 24x7 Scheduler components. This can be used to build complex distributed jobs. This can be also used for synchronization of jobs running simultaneously on different computers.

---

**See also:**

Dim  
Set  
JobRemoteRun  
GetRemoteVariable

---

## Internet statements

---

### Ping

**Description:** Checks whether the specified remote computer is online. It simulates PING utility that used to test and debug a network by sending out a data packet and waiting for a response.

**Syntax:** [Ping computer, return](#)

| Argument                 | Description   |
|--------------------------|---|
| <a href="#">Computer</a> | A string whose value is either name of the remote computer as specified in your DNS Server (for example, <a href="#">www.softtreotech.com</a> , <a href="#">ftp.softtreotech.com</a> ) or the IP number of the remote computer in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| <a href="#">Return</a>   | A boolean variable that receives the returned value   |

**Return value:** Boolean. Returns TRUE if the remote [computer](#) can be found (alive), and FALSE otherwise.

**Usage:** Use [Ping](#) to test TCP/IP connectivity and check for availability of a remote computer before making connection to that computer. You can also use Ping in various monitoring jobs of "Server Alive" kind.



**Important Notes:**

- [Ping](#) statement is available only if you have Winsock 2 installed on your system. To verify that check if you have WS2\_32.DLL in your WINDONS\SYSTEM or WINNT\SYSTEM32 directory. Most Windows 98/NT/2000 should have Winsock 2 files installed by default. If you are running Windows 95 and don't have Winsock 2 installed on your system you can freely obtain it at the following location <http://www.microsoft.com/windows/downloads/bin/W95ws2setup.exe>.
- [Ping](#) statement is available only if the TCP/IP protocol has been installed.



**Tips:**

- Turn of **Tracing** feature to see additional status messages returned by the [Ping](#) statement.
- If you see "Request timed out" message in the trace, verify that the host IP address is correct, that the host is operational, and that all the gateways (routers) between this computer and the host are operational.
- To test host name resolution by using the [Ping](#) statement, [Ping](#) the desired host using its host name. If the [Ping](#) fails with an "Unknown host" message in the trace, verify that the host name is correct and that the host name can be resolved by your DNS server.

**See also:**

PingPort  
FTP Statements  
Internet Statements

---

## PingPort

**Description:** Checks whether the specified port (e.g. network service) is responsive on the specified remote computer.

**Syntax:** [PingPort](#) computer, port, return

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Computer</a> | A string whose value is either name of the remote computer as specified in your DNS Server (for example, <a href="http://www.softtreotech.com">www.softtreotech.com</a> , <a href="ftp://ftp.softtreotech.com">ftp.softtreotech.com</a> ) or the IP number of the remote computer in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| <a href="#">Port</a>     | A number whose value is the port number that you want to check.  |
| <a href="#">Return</a>   | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns TRUE if the [Port](#) on remote [computer](#) is responding (alive), and FALSE otherwise.

**Usage:** Use [PingPort](#) to test various network services (for example, FTP) before using them. You can also use [PingPort](#) in various monitoring jobs of "Service Alive" kind.

The following table lists default port numbers for most commonly used network services.

| Service Name | Port |
|--------------|------|
| daytime      | 13   |
| netstat      | 15   |
| FTP          | 21   |
| telnet       | 23   |
| SMTP         | 25   |
| DNS          | 53   |
| finger       | 79   |

|               |     |
|---------------|-----|
| HTTP          | 80  |
| rlogin        | 513 |
| rsh           | 514 |
| UUCP          | 540 |
| klogin        | 543 |
| krcmd, kshell | 544 |

**Important Notes:**

- **PingPort** statement is available only if you have Winsock 2 installed on your system. To verify that check if you have WS2\_32.DLL in your WINDONS\SYSTEM or WINNT\SYSTEM32 directory. Most Windows 98/NT/2000 should have Winsock 2 files installed by default. If you are running Windows 95 and don't have Winsock 2 installed on your system you can freely obtain it at the following location <http://www.microsoft.com/windows/downloads/bin/W95ws2setup.exe>.
- **PingPort** statement is available only if the TCP/IP protocol has been installed.

**Tips:**

- Turn of **Tracing** feature to see additional status messages returned by the **Ping** statement.
- If you see "Request timed out" message in the trace, verify that the host IP address is correct, that the host is operational, and that all the gateways (routers) between this computer and the host are operational.
- To test host name resolution by using the **Ping** statement, **Ping** the desired host using its host name. If the **Ping** fails with an "Unknown host" message in the trace, verify that the host name is correct and that the host name can be resolved by your DNS server.

**See also:**

Ping

FTP Statements

Internet Statements

---

## WebConfig

**Description:** Set various parameters for the subsequent Internet operations.

**Syntax:** **WebConfig** *property*, *new\_value*

| Argument         | Description   |
|------------------|---|
| <b>Property</b>  | A string whose value is the name of the property for the Internet session that you want to change. The following properties are supported: <ul style="list-style-type: none"><li>• <b>"PROXY"</b></li></ul> |
| <b>New_value</b> | <ul style="list-style-type: none"><li>• A string whose value is the new value for the <b>property</b> that you want to change.</li></ul>  |

**Return value:** None.

**Usage:** Use [WebConfig](#) statement to configure your Internet connection before accessing file on the Internet using [WebGetFile](#) and [WebPostData](#) statements. Both parameter name and new value are case insensitive.

"[PROXY](#)" setting consists of 4 parts separated by commas: proxy server name (or IP address), proxy server port, your user ID and password required for proxy authentication. If your proxy server does not require authentication, do not specify user ID and password. If you use different proxies for different protocols such as HTTP and FTP, call [WebConfig](#) with different parameters before using [WebGetFile](#) and [WebPostData](#) statements for a different protocol.

**Example:**

[WebConfig](#) "[PROXY](#)", "[127.01.01.3,8080,myname,mypassword](#)"

**See also:**

[WebPostData](#)  
[WebGetFile](#)

---

## WebGetFile

**Description:** Downloads file from the specified URL. Downloaded file can be HTML file or a file of any other type.

**Syntax:** [WebGetFile](#) url, file

| Argument             | Description  |
|----------------------|--|
| <a href="#">URL</a>  | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) that points to the file that you want to download |
| <a href="#">File</a> | A string whose value is the name of the local file in which you want to save returned data   |

**Return value:** None.

**Usage:** The specified [URL](#) can point to the file of any downloadable type, including HTML files, ASCII files, image files, and other binary files. If the specified [URL](#) points to a Internet server executable file such as CGI, ASP, DLL, and other, [WebGetFile](#) returns the data source that is created as a result of processing on the Internet server. You can use [WebGetFile](#) statement for downloading various Internet files.

If the web site specified by [URL](#) requires user authentication, use the following [URL](#) syntax:

<username>:<password>@<protocol>://<server name>/<...path and file name>

Example: [myname:mypassword@http://www.mycompany.com/webstats/access.log](#)

If you connect to the Internet via proxy server, use [WebConfig](#) statement to specify connection parameters required by your proxy server.



**Note:**

- [WebGetFile](#) statement can use HTTP and FTP protocols when accessing files on the Web. If you omit protocol name in the [URL](#) specification then HTTP protocol is used by default.

**See also:**

[WebConfig](#)

WebPostData  
FTPGetFile  
WebOpenPage

---

## WebPostData

**Description:** Performs an HTTP Post, allowing a job to send a request through CGI, NSAPI, or ISAPI.

**Syntax:** [WebPostData url, data, file](#)

| Argument             | Description  |
|----------------------|--|
| <a href="#">URL</a>  | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) to post |
| <a href="#">Data</a> | A string whose value is the data to be posted to the specified <a href="#">URL</a>                       |
| <a href="#">File</a> | A string whose value is the name of the local file in which you want to save returned data               |

**Return value:** None.

**Usage:** Use this statement to invoke a CGI, NSAPI, or ISAPI function. The data returned after the POST operation can be of any downloadable type, including HTML files, ASCII files, image files, and other binary files.

If the web site specified by [URL](#) requires user authentication, use the following [URL](#) syntax:

<username>:<password>@<protocol>://<server name>/<...path and file name>

Example: [myname:mypassword@http://www.mycompany.com/cgi-bin/input.cgi](#)

If you connect to the Internet via proxy server, use [WebConfig](#) statement to specify connection parameters required by your proxy server.

**See also:**

WebConfig  
WebGetFile  
FTPPutFile  
WebOpenPage

---

## WebGetPageHTML

**Description:** Retrieves page source for the specified URL. Page source can be HTML file or a file of other type. This statement is provided for compatibility with previous versions of 24x7 Scheduler. New 24x7 Scheduler jobs should use the more powerful **WebGetFile** statement.

**Syntax:** [WebGetPageHTML url, file](#)

| Argument             | Description  |
|----------------------|--|
| <a href="#">url</a>  | A string whose value is the URL whose source data you want to retrieve                     |
| <a href="#">file</a> | A string whose value is the name of the local file in which you want to save returned data |

**Return value:** None.

**Usage:** The specified URL can point to the file of any downloadable type, including HTML files, ASCII files, image files, and other binary files. If the specified URL points to a Internet server executable file such as CGI, ASP, DLL, and other, [WebGetPageHTML](#) returns the data source that is created as a result of processing on the Internet server. You can use [WebGetPageHTML](#) statement for downloading various Internet files.



**Note:**

- [WebGetPageHTML](#) statement uses HTTP protocol for any type of file transfer.
- [WebGetPageHTML](#) statement does not support proxy connections. It is provided only for compatibility with previous versions of 24x7 Scheduler. New 24x7 Scheduler jobs should use the more powerful **WebGetFile** statement.

**See also:**

FTPGetFile  
WebOpenPage

---

## WebOpenPage

**Description:** Opens the default Web browser, displaying the specified URL.

**Syntax:** [WebOpenPage url](#)

| Argument            | Description   |
|---------------------|---|
| <a href="#">url</a> | A string whose value is the URL you want to open in the default browser |

**Return value:** None.

**See also:**

[WebGetPageHTML](#)

---

## Logical statements

---

### And

**Description:** Obtains the result of logical operation [AND](#).

**Syntax:** [And](#) [boolean1](#), [boolean2](#), [return](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">boolean1</a> | A boolean value you want to compare with value in <a href="#">boolean2</a> |
| <a href="#">boolean2</a> | A boolean value you want to compare with value in <a href="#">boolean1</a> |
| <a href="#">return</a>   | A boolean variable that receives the returned value                        |

**Return value:** Boolean. Returns result of logical AND operation. [boolean1](#) AND [boolean2](#) is true if both are true. [boolean1](#) AND [boolean2](#) is false if either is false. See the table below for returned values.

| Boolean1 | Boolean2 | Result |
|----------|----------|--------|
| TRUE     | TRUE     | TRUE   |
| FALSE    | FALSE    | FALSE  |
| TRUE     | FALSE    | FALSE  |
| FALSE    | TRUE     | FALSE  |

---

### IsDate

**Description:** Tests whether a string value is a valid date.

**Syntax:** [IsDate](#) ( [string](#), [result](#) )

| Argument               | Description   |
|------------------------|---|
| <a href="#">String</a> | A string whose value you want to test to determine whether it is a valid date |
| <a href="#">result</a> | A boolean variable that receives the returned value                           |

**Return value:** Boolean. Returns TRUE if [string](#) is a valid date and FALSE if it is not.



**Note:**

Valid dates in strings can include any combination of day (1 to 31), month (1 to 12 or the name or abbreviation of a month), and year (2 or 4 digits). 24x7 Scheduler assumes a 4-digit number is a year. Leading zeros are optional for month and day. The month, whether a name, an abbreviation, or a number, must be in the month location specified in

the system setting for a date's format. If you do not know the system setting, use the standard data type date format yyyy-mm-dd.

---

## IsDateBetween

**Description:** Tests whether a date value is between specified start and end dates.

**Syntax:** `IsDateBetween ( testdate, startdate, enddate, result )`

| Argument               | Description  |
|------------------------|--|
| <code>testdate</code>  | A date whose value you want to test to determine whether it is between <code>startdate</code> and <code>enddate</code> |
| <code>startdate</code> | A date whose value is lower limit of the specified time interval   |
| <code>enddate</code>   | A date whose value is upper limit of the specified time interval   |
| <code>result</code>    | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns TRUE if `testdate` is equal or later than `startdate` and equal or earlier than `enddate`, and returns FALSE if it is not.

---

## IsEqual

**Description:** Compares two values and returns True if the first value is equal the second one.

**Syntax:** `IsEqual a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a = b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other.

**See also:**

- IsEqual
- IsGreater
- IsGreaterOrEqual
- IsLess
- IsLessOrEqual

---

## IsGreater

**Description:** Compares two values and returns True if the first value is greater then the second one.

**Syntax:** `IsGreater a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a > b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other. All data types supported except `boolean`.

**See also:**

- `IsEqual`
- `IsGreater`
- `IsGreaterOrEqual`
- `IsLess`
- `IsLessOrEqual`

---

## IsGreaterOrEqual

**Description:** Compares two values and returns True if the first values is greater or equal the second one.

**Syntax:** `IsGreaterOrEqual a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a >= b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other. All data types supported except `boolean`.

**See also:**

- `IsEqual`

IsGreater  
IsGreaterOrEqual  
IsLess  
IsLessOrEqual

---

## IsHoliday

**Description:** Tests whether the specified date falls on a holiday.

**Syntax:** `IsHoliday ( testdate, result )`

| Argument              | Description  |
|-----------------------|--|
| <code>testdate</code> | A date whose value you want to test to determine whether it is a holiday |
| <code>result</code>   | A boolean variable that receives the returned value                      |

**Return value:** Boolean. Returns TRUE if `testdate` is a holiday and FALSE if it is not.



**Note:**

You use Tools/Holidays menu command to maintain list of holidays and other exception dates for the 24x7 Scheduler.

---

## IsLess

**Description:** Compares two values and returns True if the first value is less then the second one.

**Syntax:** `IsLess a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a < b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other. All data types supported except `boolean`.

**See also:**

IsEqual  
IsGreater  
IsGreaterOrEqual

IsLess  
IsLessOrEqual

---

## IsLessOrEqual

**Description:** Compares two values and returns True if the first value is less or equal the second one.

**Syntax:** `IsLessOrEqual a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a <= b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other. All data types supported except `boolean`.

**See also:**

IsEqual  
IsGreater  
IsGreaterOrEqual  
IsLess  
IsLessOrEqual

---

## IsNumber

**Description:** Reports whether the value of a string is a number.

**Syntax:** `IsNumber ( string, result )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | A string whose value you want to test to determine whether it is a valid number |
| <code>result</code> | A boolean variable that receives the returned value                             |

**Return value:** Boolean. Returns TRUE if `string` is a valid number and FALSE if it is not.

## IsTime

**Description:** Reports whether the value of a string is a valid time value.

**Syntax:** `IsTime (timevalue, result )`

| Argument               | Description   |
|------------------------|---|
| <code>timevalue</code> | A string whose value you want to test to determine whether it is a valid time |
| <code>result</code>    | A boolean variable that receives the returned value                           |

**Return value:** Boolean. Returns TRUE if `timevalue` is a valid time and FALSE if it is not.

---

## IsTimeBetween

**Description:** Tests whether a time value is between specified start and end times.

**Syntax:** `IsTimeBetween ( testtime, starttime, endtime, result )`

| Argument               | Description  |
|------------------------|--|
| <code>testdate</code>  | A time whose value you want to test to determine whether it is between <code>starttime</code> and <code>endtime</code> |
| <code>starttime</code> | A time whose value is lower limit of the specified time interval   |
| <code>endtime</code>   | A time whose value is upper limit of the specified time interval   |
| <code>result</code>    | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns TRUE if `testtime` is equal or later than `starttime` and equal or earlier than `endtime`, and returns FALSE if it is not.

---

## IsWeekday

**Description:** Tests whether the specified date falls on a weekday.

**Syntax:** `IsWeekday ( testdate, result )`

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Testdate</a> | A date whose value you want to test to determine whether it is a weekday |
| <a href="#">result</a>   | A boolean variable that receives the returned value                      |

**Return value:** Boolean. Returns TRUE if [testdate](#) is a weekday and FALSE if it is not.

**Note:**

Weekdays are days from Monday through Friday.

---

## IsWeekend

**Description:** Tests whether the specified date falls on a weekend.

**Syntax:** [IsWeekend](#) ( [testdate](#), [result](#) )

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Testdate</a> | A date whose value you want to test to determine whether it is a weekend |
| <a href="#">result</a>   | A boolean variable that receives the returned value                      |

**Return value:** Boolean. Returns TRUE if [testdate](#) is a weekend and FALSE if it is not.

**Note:**

Weekends include the 2 following days: Saturday and Monday.

---

## Not

**Description:** Obtains the result of logical operation [NOT](#).

**Syntax:** [Not](#) [boolean](#), [return](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">boolean</a> | The boolean value you want to negate                |
| <a href="#">return</a>  | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns result of logical negation operation. See the table below for returned values.

| Boolean | Result |
|---------|--------|
| TRUE    | FALSE  |
| FALSE   | TRUE   |

## NotEqual

**Description:** Compares two values and returns True if the first value is not equal the second one.

**Syntax:** `NotEqual a, b, return`

| Argument            | Description   |
|---------------------|---|
| <code>a</code>      | A value that you want to compare with <code>b</code>        |
| <code>b</code>      | A value against which you want compare value <code>a</code> |
| <code>return</code> | A boolean variable that receives the returned value         |

**Return value:** Boolean. Returns TRUE when `a <> b`, and FALSE otherwise

**Usage:** The data types of value `a` and `b` must match each other.

**See also:**

- IsEqual
- IsGreater
- IsGreaterOrEqual
- IsLess
- IsLessOrEqual

---

## Or

**Description:** Obtains the result of logical operation OR.

**Syntax:** `Or boolean1, boolean2, return`

| Argument              | Description   |
|-----------------------|---|
| <code>boolean1</code> | A boolean value you want to compare with value in <code>boolean2</code> |
| <code>boolean2</code> | A boolean value you want to compare with value in <code>boolean1</code> |
| <code>return</code>   | A boolean variable that receives the returned value                     |

**Return value:** Boolean. Returns result of logical OR operation. `boolean1` OR `boolean2` is true if either is true or both are true. `boolean1` OR `boolean2` is false only if both are false. See the table below for returned values.

| Boolean1 | Boolean2 | Result |
|----------|----------|--------|
| TRUE     | TRUE     | TRUE   |
| FALSE    | FALSE    | FALSE  |

|       |       |      |
|-------|-------|------|
| TRUE  | FALSE | TRUE |
| FALSE | TRUE  | TRUE |

---

## Mail statements

---

### MailSend

**Description:** Establishes a new mail session and sends the specified mail message. If the message information is incomplete, the mail system displays a dialog box that you use to enter missing information.

**Syntax:** `MailSend profile, password, recipient, subject, message`

| Argument               | Description   |
|------------------------|---|
| <code>profile</code>   | The <code>profile</code> value depends on the email system interface that you have selected in the 24x7 Scheduler Options.<br><br>MAPI: a string whose value is the profile name to use when starting a new MAPI session<br><br>Lotus Notes: a string whose value is the user name to use when logging to Lotus Notes<br><br>SMTP: a string whose value is the valid sender's email address |
| <code>password</code>  | A string whose value is the user's mail system password   |
| <code>recipient</code> | A string variable whose value is the name of the recipient for the message.   |
| <code>subject</code>   | A string variable whose value is the subject line, displayed in the message header  |
| <code>message</code>   | A string variable whose value is the content of the message body  |

**Return value:** None.

**Usage:** If one or more message parameters are missing or incorrect, the mail system displays a dialog box. Execution of the script will not continue until you either enter missing information interactively or cancel the dialog. You can specify multiple recipients in one message. Use comma (,) to separate recipient names.

**See also:**

MailSendWithAttachment

## MailSendWithAttachment

**Description:** Establishes a new mail session and sends a mail message. If no message information is supplied, the mail system provides a dialog box for entering it before sending the message. An additional file can be attached to the message.

**Syntax:** `MailSendWithAttachment profile, password, recipient, subject, message, file`

| Argument               | Description   |
|------------------------|---|
| <code>profile</code>   | The <code>profile</code> value depends on the email system interface that you have selected in the 24x7 Scheduler Options.<br><br>MAPI: a string whose value is the profile name to use when starting a new MAPI session<br><br>Lotus Notes: a string whose value is the user name to use when logging to Lotus Notes<br><br>SMTP: a string whose value is the valid sender's email address |
| <code>password</code>  | A string whose value is the user's mail system password   |
| <code>recipient</code> | A string variable whose value is the name of the recipient for the message.   |
| <code>subject</code>   | A string variable whose value is the subject line, displayed in the message header  |
| <code>message</code>   | A string variable whose value is the content of the message body  |
| <code>file</code>      | A string variable whose value is the full file name of the file you want to attach  |

**Return value:** None.

**Usage:** If one or more message parameters are missing or incorrect, the mail system displays a dialog box. Execution of the script will not continue until you either enter missing information interactively or cancel the dialog. You can specify multiple recipients in one message. Use comma (,) to separate recipient names. You can also specify multiple attachments in one message. Use comma (,) to separate attachment file names.

**See also:**

MailSend

---

## Miscellaneous statements

---

### AgentTest

**Description:** Tests availability of the specified 24x7 Remote Agent.

**Syntax:** `AgentTest agent, return`

| Argument            | Description  |
|---------------------|--|
| <code>Agent</code>  | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile. |
| <code>Return</code> | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns `TRUE` if the specified `Agent` is running and successful connection can be established between the main scheduler and this Remote Agent, otherwise returns `FALSE`.

**Usage:** Use this statement to check availability of the Remote Agent before executing `JobRemoteRun` statement or `JobRun` statement for a job that should be executed remotely by that Agent. You can also use that statement to check whether the remote computer hosting 24x7 Remote Agent is running.

**See also:**

- JobRun
- JobRemoteRun
- Remote Agents

---

## Call

**Description:** Calls a function within a dynamic-link library or an executable file.

**Syntax:** `Call ( module, function, parameter-specification, has-result, [arguments], [result] )`

| Argument                             | Description   |
|--------------------------------------|---|
| <code>Module</code>                  | A string whose value is the filename of the DLL or EXE.   |
| <code>Function</code>                | A string whose value is the name of the function to run in the designated <code>module</code> .   |
| <code>Parameter-specification</code> | A string whose value is indicating the formats of parameters passed to the <code>function</code> . Characters in the string represent C parameter types: Characters in upper case indicate that the corresponding parameters must be passed by reference, characters in lower case indicate that the corresponding parameters must be passed by value. Parameters of string and double data |

|                                       |   |
|---------------------------------------|---|
|                                       | types are always passed by reference.<br><b>Character</b> <b>Description</b><br><a href="#">U</a> or <a href="#">u</a> unsigned integer<br><a href="#">L</a> or <a href="#">l</a> long<br><a href="#">N</a> or <a href="#">n</a> short<br><a href="#">C</a> or <a href="#">c</a> byte or char<br><a href="#">F</a> or <a href="#">f</a> float<br><a href="#">D</a> or <a href="#">d</a> double (always by ref)<br><a href="#">S</a> or <a href="#">s</a> string (always by ref) |
| <a href="#">Has-result</a>            | A boolean whose value is indicating whether the calling function returns a result.  |
| <a href="#">Parameters (optional)</a> | A list of parameters that must be passed to the <a href="#">function</a> . Order, number, and data types of parameters must much <a href="#">parameter-specification</a>  |
| <a href="#">result (optional)</a>     | A variable that receives the returned value. You must specify data type of the <a href="#">result</a> value as the last character in the <a href="#">parameter-specification</a> .  |

**Return value:** See parameter description above.

**Usage:** It is recommended that you include the .DLL filename extension when specifying the name of the DLL. Without the extension, 24x7 Scheduler first looks through all the system search paths directories trying to find the file exactly as specified. Only after this fails does it add the .DLL extension and look through all the search paths again.

24x7 Scheduler supports only functions and subroutines in 32-bit DLLs and EXEs that use the [\\_stdcall](#) calling convention.  
See examples for more information on usage.

**See also:**

String  
Format

---

## CD

**Description:** Changes the current directory for the current process.

**Syntax:** [CD dir](#)

| Argument            | Description  |
|---------------------|--|
| <a href="#">dir</a> | A string that is the directory you want to set as current directory. |

**Usage:** The [dir](#) specifies the path to the new current directory. This parameter may be a relative path or a fully qualified path. In either case, the fully qualified path of the specified directory is calculated and stored as the current directory. This new setting will affect all file operations that do not specify full paths to the referenced files.



**Notes:**

Each process has a single current directory made up of two parts:

- A disk designator that is either a drive letter followed by a colon, or a server name and share name (\\servername\sharename)
- A directory on the disk designator

---

## DiskGetFreeSpace

**Description:** Retrieves the amount of free space on the specified disk.

**Syntax:** [DiskGetFreeSpace](#) path, freespace

| Argument                  | Description   |
|---------------------------|---|
| <a href="#">Path</a>      | A string that is the disk letter you want to query or a file name or file path. In last two cases <a href="#">DiskGetFreeSpace</a> uses the first character only. |
| <a href="#">Freespace</a> | A numeric variable that receives the returned value   |

**Usage:** The [freespace](#) receives the amount of free space in bytes.



**Important notes:**

- **Windows 95:** The [DiskGetFreeSpace](#) statement returns incorrect value for volumes that are larger than 2 gigabytes. That is because the operating system manipulates the disk properties so that computations with them yield the incorrect volume size.
- **Windows 95 OSR 2, Windows 98, and Windows NT:** The [DiskGetFreeSpaceEx](#) statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2), Windows 98, and Windows NT. The [GetDiskFreeSpaceEx](#) statement returns correct values for all volumes, including those that are greater than 2 gigabytes.

**See also:**

DiskGetFreeSpaceEx  
MemoryGetFree

---

## DiskGetFreeSpaceEx

**Description:** Retrieves the amount of free space on the specified disk.

**Syntax:** [DiskGetFreeSpaceEx](#) path, freespace

| Argument                  | Description   |
|---------------------------|---|
| <a href="#">path</a>      | A string that is the disk letter you want to query or a file name or file path. In last two cases <a href="#">DiskGetFreeSpaceEx</a> uses the first character only. |
| <a href="#">freespace</a> | A numeric variable that receives the returned value   |

**Usage:** The [freespace](#) receives the amount of free space in bytes.



**Important notes:**

- **Windows 95:** The [DiskGetFreeSpaceEx](#) statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2) only.  
The [DiskGetFreeSpace](#) statement returns incorrect value for volumes that are larger than 2 gigabytes. That is because the operating system manipulates the disk properties so that computations with them yield the incorrect volume size.
- **Windows 95 OSR 2, Windows 98, and Windows NT:** The [DiskGetFreeSpaceEx](#) statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2), Windows 98, and Windows NT. The [GetDiskFreeSpaceEx](#) statement returns correct value for all volumes, including those that are greater than 2 gigabytes.

**See also:**

DiskGetFreeSpace  
MemoryGetFree

---

## LogAddMessage

**Description:** Writes an entry at the end of the Windows NT application event log..

**Syntax:** [LogAddMessage severity, message](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">severity</a> | A string that is the severity of the message you want to add to the system application event log. It can be one of the following: <ul style="list-style-type: none"><li>• "ERROR"</li><li>• "WARNING"</li><li>• "INFO"</li></ul> |
| <a href="#">message</a>  | A string that is the message you want to add to the system application event log.  |

**Usage:** This statement can be used on Windows NT/2000 systems only. You can use the Windows NT System Event Viewer to read event log messages.



**Note:**

The 24x7 Scheduler maintains its own event log that is different from the system event log. LogAddMessage does not affect 24x7 Scheduler event log.

**See also:**

LogAddMessageEx  
LogSearch

## LogAddMessageEx

**Description:** Adds new record to the 24x7 Scheduler event log.

**Syntax:** [LogAddMessageEx severity, job\\_id, job\\_name, message](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">severity</a> | A string that is the severity of the message you want to add to the 24x7 Scheduler event log. It can be one of the following: <ul style="list-style-type: none"><li>• "ERROR"</li><li>• "WARNING"</li><li>• "INFO"</li></ul> |
| <a href="#">job_id</a>   | A number whose value should be either ID of the job to which this message belongs or 0 to indicate that the message has global scope.  |
| <a href="#">job_name</a> | A string whose value should be either name of the job to which this message belongs or some generic name like "24x7 Scheduler" to indicate that the message has global scope.  |
| <a href="#">message</a>  | A string that is the message you want to add to the 24x7 Scheduler event log.  |

**Usage:** [LogAddMessageEx](#). writes new record to the 24x7 Scheduler event log. If parallel logging to the Windows NT event log enabled, [LogAddMessageEx](#). also writes an entry at the end of the Windows NT application event log.



**Note:**

24x7 Scheduler does not verify [job\\_id](#) and [job\\_name](#) values. This is your responsibility to specify the correct values.

**See also:**

[LogAddMessage](#)  
[LogSearch](#)

---

## LogSearch

**Description:** Search entries in the Windows NT event log.

**Syntax:** [LogSearch search\\_system, search\\_warnings, start\\_time, return](#)

| Argument                        | Description  |
|---------------------------------|--|
| <a href="#">search_system</a>   | A boolean that indicates whether you want to search the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log. |
| <a href="#">search_warnings</a> | A boolean that indicates whether you want to search for messages having severity either ERROR or WARNING.  |

|                            |   |
|----------------------------|---|
|                            | Specify TRUE to search for both types or FALSE to search ERRORS only.   |
| <a href="#">start_time</a> | A date-time value that is the earliest time of the messages that you want to search. Use this parameter as a “filter” for the event log. 24x7 Scheduler will search only messages that were logged since the <a href="#">start_time</a> . |
| <a href="#">Return</a>     | A string variable that receives the returned value.   |

**Return value:** String. Returns all found messages as a tab-separated multi-line string. Each line consists of message time, message ID, message severity (either ERROR or WARNING), message source, user name, and message text separated by the tab character.

**Usage:** This statement can be used on Windows NT systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to search Windows NT event logs for reported errors and this way effectively monitor other application that write to the event logs.

**Note:**

The 24x7 Scheduler maintains its own event log that is different from the system event log. [LogSearch](#) does not search 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT System Event Log then [LogSearch](#) statement will also automatically search 24x7 Scheduler generated messages.

**See also:**

[LogAddMessage](#)

---

## LogSearchEx

**Description:** Search entries in the Windows NT event log written by the specified application.

**Syntax:** [LogSearchEx](#) [search\\_system](#), [start\\_time](#), [event\\_source](#), [return](#)

| Argument                      | Description   |
|-------------------------------|---|
| <a href="#">search_system</a> | A boolean that indicates whether you want to search the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log.  |
| <a href="#">start_time</a>    | A date-time value that is the earliest time of the messages that you want to search. Use this parameter as a “filter” for the event log. 24x7 Scheduler will search only messages that were logged since the <a href="#">start_time</a> . |
| <a href="#">event_source</a>  | A string whose value is the name of the event source (e.g. name of the application that wrote the message).   |
| <a href="#">Return</a>        | A string variable that receives the returned value.   |

**Return value:** String. Returns all found messages as a tab-separated multi-line string. Each line consists of message time, message ID, message severity (either INFO, ERROR or WARNING), message source, user name, and message text separated by the tab character.

**Usage:** This statement can be used on Windows NT systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to search Windows NT event logs for reported errors and this way effectively monitor other application that write to the event logs.

**Note:**

The 24x7 Scheduler maintains its own event log that is different from the system event log. [LogSearchEx](#) does not search 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT System Event Log then [LogSearchEx](#) statement will also automatically search 24x7 Scheduler generated messages.

**See also:**

[LogAddMessage](#)  
[LogSearch](#)  
[LogWaitForUpdate](#)

---

## LogWaitForUpdate

**Description:** Suspends job execution and then enters an efficient wait state until either new event is written to the Windows NT event log or the statement times out.

**Syntax:** [LogWaitForUpdate](#) [wait\\_system](#), [event\\_source](#), [timeout](#), [return\\_data](#), [return\\_status](#)

| Argument                      | Description   |
|-------------------------------|---|
| <a href="#">wait_system</a>   | A boolean that indicates whether you want to wait for the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log.  |
| <a href="#">event_source</a>  | A string whose value is the name of the event source (e.g. name of the application that wrote the message). If you specify an empty string "", 24x7 Scheduler will resume after any new event record is added to the event log, otherwise it will resume only after a record is added by the specified <a href="#">event_source</a> . |
| <a href="#">timeout</a>       | A number whose value is the maximum time interval within which you expect a new event record to be written to the specified event log. Use 0 timeout to allow infinite waiting.   |
| <a href="#">return_data</a>   | A string variable that receives the returned event record data.   |
| <a href="#">return_status</a> | A boolean variable that receives the returned status value.   |

**Return value:** String and Boolean. If the [LogWaitForUpdate](#) statement succeeds, the [return\\_status](#) is [True](#) and the [return\\_data](#) value contains new event log record as a tab-separated string, which consists of message time, message ID, message severity (either INFO, ERROR or WARNING), message source, user name, and message text. If the [LogWaitForUpdate](#) statement fails, the [return\\_status](#) value is [False](#) and [return\\_data](#) is empty string "".

**Usage:** This statement can be used on Windows NT systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to monitor Windows NT event logs for new messages.

**Note:**

The 24x7 Scheduler maintains its own event log that is different from the system event log. [LogWaitForUpdate](#) does not monitor 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT System Event Log then [LogWaitForUpdate](#) statement will also automatically monitor 24x7 Scheduler generated messages.

**See also:**

[LogAddMessage](#)  
[LogSearch](#)  
[LogSearchEx](#)

---

## MemoryGetFree

**Description:** Retrieves the amount of available free virtual memory.

**Syntax:** [MemoryGetFree freebytes](#)

| Argument                  | Description   |
|---------------------------|---|
| <a href="#">freebytes</a> | A numeric variable that receives the returned value |

**Usage:** The [freebytes](#) receives the amount of free virtual memory. Divide the returned value by 1024 to calculate free memory in Kbytes. divide the returned value by 1048576 to calculate free memory in Mbytes.

**Note:**

24x7 Scheduler allocates some additional memory while executing JAL script. After execution, this memory is released back to the Operation System. Therefore, amount of free memory reported by the [MemoryGetFree](#) statement is usually less than amount of free memory available after the script run.

**See also:**

[DiskGetFreeSpace](#)

---

## MessageBox

**Description:** Displays a system Message Box with the text you specified, and YES/NO buttons.

**Syntax:** [MessageBox message](#)

| Argument                | Description                                      |
|-------------------------|--|
| <a href="#">message</a> | A string that is the message you want to display |

**Usage:** The [MessageBox](#) displays standard Windows message box with the user-defined text and the prompt to press YES button to continue or press NO to stop the script. Pressing the NO button interrupts the script execution.

You should use this statement for Debugging purposes only. For instance, you can use it to display the value stored in a variable.

---

## Reboot

**Description:** Immediately logs off, shuts down, and restarts the system ("cold" reboot.).

**Syntax:** [Reboot](#)

**Usage:** This statement has no arguments.



**Note:**

During a shutdown operation, applications that are shut down are allowed a specific amount of time to respond to the shutdown request. If the time expires, Windows forces applications to close.

---

## Yield

**Description:** Yields execution so that the operating system can process other events.

**Syntax:** [Yield](#)

**Usage:** This statement has no arguments.

---

## Numeric statements

---

### Add

**Description:** Obtains the result of an addition operation.

**Syntax:** [Add](#) [n1](#), [n2](#), [sum](#)

| Argument            | Description  |
|---------------------|--|
| <a href="#">n1</a>  | The number to which you want to add <a href="#">n2</a> . |
| <a href="#">n2</a>  | The number you want to add to <a href="#">n1</a>         |
| <a href="#">sum</a> | A numeric variable that receives the returned value      |

**Return value:** Returns the sum of [n1](#) and [n2](#) so that [sum](#) = [n1](#) + [n2](#)

**Usage:** You can also use keyword [Plus](#) instead of [Add](#).

**See also:**

Subtract  
Multiply  
Divide  
Mod

---

## Ceiling

**Description:** Determines the smallest whole number that is greater than or equal to a specified limit.

**Syntax:** [Ceiling n, return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">n</a>      | The number for which you want the smallest whole number that is greater than or equal to it |
| <a href="#">return</a> | A numeric variable that receives the returned value   |

**Return value:** Returns the smallest whole number that is greater than or equal to [n](#).

**See also:**

Floor

---

## Divide

**Description:** Obtains the result of a division operation.

**Syntax:** [Divide n1, n2, div](#)

| Argument            | Description   |
|---------------------|---|
| <a href="#">n1</a>  | The number you want to divide by <a href="#">n2</a>   |
| <a href="#">n2</a>  | The number you want to divide into <a href="#">n1</a> |
| <a href="#">div</a> | A numeric variable that receives the returned value   |

**Return value:** Returns the result of division [n1](#) by [n2](#) so that [div](#) = [n1](#) / [n2](#)

**See also:**

Subtract  
Multiply  
Add

Mod

---

## Floor

**Description:** Determines the largest whole number less than or equal to a number.

**Syntax:** `Floor n, return`

| Argument            | Description   |
|---------------------|---|
| <code>n</code>      | The number for which you want the largest whole number that is less than or equal to it |
| <code>return</code> | A numeric variable that receives the returned value                                     |

**Return value:** Returns the largest whole number less than or equal to `n`.

**See also:**

Ceiling

---

## Max

**Description:** Determines the larger of two numbers.

**Syntax:** `Max n1, n2, return`

| Argument            | Description   |
|---------------------|---|
| <code>n1</code>     | The number to which you want to compare <code>n2</code> |
| <code>n2</code>     | The number to which you want to compare <code>n1</code> |
| <code>return</code> | A numeric variable that receives the returned value     |

**See also:**

Min

---

## Min

**Description:** Determines the smaller of two numbers.

**Syntax:** `Min n1, n2, return`

| Argument               | Description  |
|------------------------|--|
| <a href="#">n1</a>     | The number to which you want to compare <a href="#">n2</a> |
| <a href="#">n2</a>     | The number to which you want to compare <a href="#">n1</a> |
| <a href="#">return</a> | A numeric variable that receives the returned value        |

**See also:**Max

---

## Mod

**Description:** Obtains the remainder (modulus) of a division operation.**Syntax:** [Mod](#) [n1](#), [n2](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">n1</a>     | The number you want to divide by <a href="#">n2</a>   |
| <a href="#">n2</a>     | The number you want to divide into <a href="#">n1</a> |
| <a href="#">return</a> | A numeric variable that receives the returned value   |

**Return value:** Returns the reminder of a division using the following formula:  $\text{return} = n2 - (\text{floor}(n2 / n1) * n1)$ **See also:**

Subtract

Multiply

Divide

Add

---

## Multiply

**Description:** Obtains the result of a multiplication operation.**Syntax:** [Multiply](#) [n1](#), [n2](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">n1</a>     | The number you want to multiply by <a href="#">n2</a>         |
| <a href="#">n2</a>     | The number you want to be a multiplier for <a href="#">n1</a> |
| <a href="#">return</a> | A numeric variable that receives the returned value           |

**Return value:** Returns the result of multiplication [n1](#) by [n2](#) so that  $\text{return} = n1 * n2$ .

**See also:**

- Subtract
- Add
- Divide
- Mod

---

## Power

**Description:** Obtains the result of an exponentiation operation.

**Syntax:** `Power n1, n2, return`

| Argument            | Description  |
|---------------------|--|
| <code>n1</code>     | The number you want to be a base for <code>n2</code>     |
| <code>n2</code>     | The number you want to be a exponent for <code>n1</code> |
| <code>return</code> | A numeric variable that receives the returned value      |

**Return value:** Returns the result of exponentiation `n1` by `n2` so that `return = n1n2`.

**See also:**

- Subtract
- Multiply
- Divide
- Mod
- Add

---

## Round

**Description:** Rounds a number to the specified number of decimal places.

**Syntax:** `Round n1, n2, return`

| Argument            | Description   |
|---------------------|---|
| <code>n1</code>     | The number you want to round  |
| <code>n2</code>     | The number of decimal places to which you want to round <code>n1</code> |
| <code>return</code> | A numeric variable that receives the returned value                     |

**Usage:** This round operation has effect only when `n1` has more decimal places then specified in `n2`.



**Tip:**

Use 0 in place of [n2](#) for rounding [n1](#) to the nearest integer number.

**See also:**

Floor  
Ceiling

---

## Subtract

**Description:** Obtains the result of a subtraction operation.

**Syntax:** [Subtract](#) [n1](#), [n2](#), [sub](#)

| Argument            | Description  |
|---------------------|--|
| <a href="#">n1</a>  | The number to which you want to add <a href="#">n2</a> . |
| <a href="#">n2</a>  | The number you want to add to <a href="#">n1</a>         |
| <a href="#">sub</a> | A numeric variable that receives the returned value      |

**Return value:** Returns the subtraction of [n1](#) and [n2](#) so that [sub](#) = [n1](#) - [n2](#)

**Usage:** You can also use keyword [Minus](#) instead of [Subtract](#).

**See also:**

Add  
Multiply  
Divide  
Mod

---

## Print statements

---

## FilePrint

**Description:** Sends the specified file to the default printer

**Syntax:** [FilePrint](#) [filename](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">filename</a> | A string whose value is the name of the file you want to print. If <code>filename</code> is not on the operating system's search path, you must enter the fully qualified name |

**Return value:** None.

**See also:**

`PrinterSetDefault`

---

## Print

**Description:** Prints the specified string.

**Syntax:** [Print s](#)

| Argument          | Description   |
|-------------------|---|
| <a href="#">s</a> | The string you want to print. If the string includes carriage return / new line character pairs (CR/NL), the string will be printed on multiple lines |

**Return value:** None.

**See also:**

`FilePrint`

`PrinterSetDefault`

---

## PrinterGetDefault

**Description:** Reports name of the default printer.

**Syntax:** [PrinterGetDefault s](#)

| Argument          | Description   |
|-------------------|---|
| <a href="#">s</a> | A string variable that receives the name of the default printer |

**Return value:** None.

**Usage:** You can use [PrinterGetDefault](#) to retrieve name of the default printer before you change it using [PrinterSetDefault](#) so that you can restore the default printer after some processing done.

**See also:**

`FilePrint`

---

PrinterSetDefault

---

## PrinterPurgeAllJobs

**Description:** Discards all pending print jobs from the specified print queue.

**Syntax:** [PrinterPurgeAllJobs s](#)

| Argument          | Description   |
|-------------------|---|
| <a href="#">s</a> | A string variable whose value is the name of a printer for which you want to discard all pending print jobs |

**Return value:** None.

**See also:**

PrinterGetDefault

PrinterSetDefault

---

## PrinterSetDefault

**Description:** Changes the default printer.

**Syntax:** [PrinterSetDefault s](#)

| Argument          | Description   |
|-------------------|---|
| <a href="#">S</a> | A string whose value is the name of the printer that you want to be a default printer |

**Return value:** None.

**See also:**

FilePrint

PrinterGetDefault

---

## Process statements

---

### IsTaskRunning

**Description:** Reports whether the specified task is running.

**Syntax:** [IsTaskRunning](#) module, return

| Argument               | Description  |
|------------------------|--|
| <a href="#">module</a> | A string whose value is the name of the main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension (.DLL) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string does not have to specify a path. The name is compared (case independently) to the names of modules currently running |
| <a href="#">return</a> | A boolean variable that receives the returned value  |

**Return value:** Boolean. Returns TRUE when the specified [module](#) is running, and FALSE otherwise.

**Usage:** You can use the Windows Task Manager to find out the [module](#) name. For example the module name for MS Word is [WINWORD.EXE](#). When you know the [module](#) name you can use [IsTaskRunning](#) statement to check state of the required application. For instance, in order to fax something, you may want to check presence of Delrina WinFax instance in a memory before establishing a DDE connection to the Delrina DDE server. If [IsTaskRunning](#) returns FALSE then you can call [Run](#) statement to open an instance of Delrina WinFax . Alternatively, you can use [ProcessList](#) statement to get a string listing all processes and their process IDs.

**See also:**

- Run
- ProcessGetID
- ProcessList
- ProcessKill
- ServiceGetStatus

---

### ProcessGetID

**Description:** Reports process ID for the specified module.

**Syntax:** [ProcessGetID](#) filename, return

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">filename</a> | A string whose value is the full name of the process main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension ( <a href="#">.DLL</a> ) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string has to include a file path to the desired module. The name is compared (case independently) to the names of modules currently running |
| <a href="#">return</a>   | A numeric variable that receives the returned value  |

**Return value:** Number. Returns Windows process identifier for the specified module. A process identifier can change for each run of the program for which you want to find process ID.

**See also:**

ProcessGetHandle  
ProcessGetWindow  
ProcessKill  
WindowGetProcess

---

## ProcessGetHandle

**Description:** Reports process handle for the specified module.

**Syntax:** [ProcessGetHandle](#) [module](#), [return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">module</a> | A string whose value is the full name of the process main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension ( <a href="#">.DLL</a> ) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string has to include a file path to the desired module. The name is compared (case independently) to the names of modules currently running |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. Returns process handle for the specified module. A process handle can change for each run of the program for which you want to find out process handle.

**Usage:** The returned handle is not global and should not be passed as a parameter to other Windows applications.

**See also:**

ProcessGetID  
ProcessGetWindow

ProcessKill  
WindowGetProcess

---

## ProcessKill

**Description:** Terminates the specified process and all of its threads.

**Syntax:** [ProcessKill processid](#)

| Argument                  | Description   |
|---------------------------|---|
| <a href="#">processid</a> | A number whose value is the id of the process which you want to terminate |

**Return value:** None.

**Usage:** The [ProcessKill](#) statement can be used to unconditionally cause a process to exit. Use it only in extreme circumstances. You can use [ProcessGetID](#) statement to find the process ID for the desired module (program). You can also use [WindowGetProcess](#) statement to get process ID for the desired window.



**Note:**

Termination of a process does not always forces termination of all its child processes!

**See also:**

ProcessGetID  
WindowGetProcess

---

## ProcessList

**Description:** Reports all running processes and their process IDs.

**Syntax:** [ProcessList return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">return</a> | A string variable that receives the returned value |

**Return value:** String. Returns Ids and names of all running process as multi-line string. Each line consists of process ID following by the tab character and process name.

**Usage:** Use the returned value to find out process name for the desired application. The application must be running at the when you call [ProcessList](#) statement.



**Note:** Because of security restrictions in Windows NT the returned string may include not all running processes.

**See also:**

ProcessGetID  
ProcessGetWindow  
ProcessKill  
WindowGetProcess

---

## ProcessGetWindow

**Description:** Finds and reports the handle of a first window whose process ID matches the specified process ID.

**Syntax:** [ProcessGetWindow processID, return](#)

| Argument                  | Description   |
|---------------------------|---|
| <a href="#">processID</a> | A number whose value is the ID of the process for which you want to find first window |
| <a href="#">return</a>    | A numeric variable that receives the returned value                                   |

**Return value:** Number. Returns the handle of the first window that Window's finds for the specified [processID](#). A process ID can change for each run of the program for which you want to find the first window.

**Usage:** The returned handle is global and uniquely identifies the window instance. You can use [ProcessGetID](#) statement to find out the ID of desired module.

**See also:**

ProcessGetHandle  
ProcessGetID  
ProcessKill  
WindowGetProcess

---

## Run

**Description:** Runs the specified program or document.

**Syntax:** [Run command, dir, return](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">command</a> | A string whose value is the full or partial path and filename of the module to execute.<br>The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a |

|                        |  |
|------------------------|--|
|                        | <p>Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer.</p> <p>If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:</p> <ol style="list-style-type: none"><li>1 The directory from which the 24x7 Scheduler loaded.</li><li>2 The current directory.</li><li>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.</li><li>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.</li><li>5 The Windows directory. The name of this directory is WINDOWS or WINNT.</li><li>6 The directories that are listed in the PATH environment variable.</li></ol> <p>You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in mind that other programs can modify the PATH environment variable as well.</p> <p>Program name can be followed by <b>command line parameters</b>. You can use <b>macro-parameters</b> inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.</p> |
| <a href="#">dir</a>    | <p>A string whose value is the directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify the empty string (""). Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document</p>  |
| <a href="#">return</a> | <p>A numeric that receives the handle of the created process.</p>  |

**Return value:** Number. Returns the handle of the created process. You can use that handle to control and manipulate further process execution.

**Usage:** You can use [Run](#) for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.

In order to run a document, its extension must be registered. For example, if you want to start a [MDB](#) file that has [AutoExec](#) macro, you must have [MDB](#) file extension registered as a MS Access database application.

**See also:**

RunAndWait  
RunAsUser  
RunWithInput  
SendKeys  
Wait

---

## RunAsUser

**Description:** Runs the specified program or document using different Windows NT user login.

**Syntax:** `RunAsUser command, dir, user, password, domain, return`

| Argument             | Description  |
|----------------------|--|
| <code>command</code> | <p>A string whose value is the full or partial path and filename of the module to execute.</p> <p>The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer.</p> <p>If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:</p> <ol style="list-style-type: none"><li>7 The directory from which the 24x7 Scheduler loaded.</li><li>8 The current directory.</li><li>9 The 32-bit Windows system directory. The name of this directory is SYSTEM32.</li><li>10 The 16-bit Windows system directory. The name of this directory is SYSTEM.</li><li>11 The Windows directory. The name of this directory is WINDOWS or WINNT.</li><li>12 The directories that are listed in the PATH environment variable.</li></ol> <p>You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the PATH environment variable as well.</p> <p>Program name can be followed by <b>command line parameters</b>. You can use <b>macro-parameters</b> inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.</p> |
| <code>dir</code>     | <p>A string whose value is the directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify the empty string (""). Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document</p>  |
| <code>user</code>    | <p>A string variable that specifies the user name. This is the name of the user account to log on to.</p>  |

|                          |   |
|--------------------------|---|
| <a href="#">password</a> | A string variable that specifies the password for the user account specified by <a href="#">user</a> .  |
| <a href="#">domain</a>   | A string variable that specifies the domain or server to log on to. If this parameter is ".", Windows NT searches only the local account database for the account specified in <a href="#">user</a> . |
| <a href="#">return</a>   | A numeric variable that receives the handle of the created process.   |

---

**Return value:** Number. Returns the handle of the created process. You can use that handle to control and manipulate further process execution.

**Usage:** You can use [RunAsUser](#) for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.

In order to run a document, its extension must be registered. For example, if you want to start a [MDB](#) file that has [AutoExec](#) macro, you must have [MDB](#) file extension registered as a MS Access database application.



**Important Notes:**

- [RunAsUser](#) statement is **not supported on Windows 95/98** systems.
- Windows NT considers the [user](#) as logged on until the process (and all child processes) have ended.
- The process created by the [RunAsUser](#) statement is non-interactive, that is, it runs on a desktop that is not visible and cannot receive user input. Also, the process inherits the environment of the 24x7 Scheduler, rather than the environment associated with the specified [user](#).

---

**See also:**

Run  
RunAsUserAndWait  
ProcessKill

---

## RunAndWait

**Description:** Starts specified program or document and enters an efficient wait state until this process finishes or until the [timeout](#) interval elapses. In the last case the 24x7 Scheduler forcibly terminates the process.

**Syntax:** [RunAndWait](#) [command](#), [dir](#), [timeout](#), [return](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">Command</a> | A string whose value is the full or partial path and filename of the module to execute.<br>The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer. |

|         |   |
|---------|---|
|         | <p>If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:</p> <ol style="list-style-type: none"> <li>1 The directory from which the 24x7 Scheduler loaded.</li> <li>2 The current directory.</li> <li>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.</li> <li>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.</li> <li>5 The Windows directory. The name of this directory is WINDOWS or WINNT.</li> <li>6 The directories that are listed in the PATH environment variable.</li> </ol> <p>You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in mind that other programs can modify the PATH environment variable as well.</p> <p>Program name can be followed by <b>command line parameters</b>. You can use <b>macro-parameters</b> inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.</p> |
| Dir     | <p>A string whose value is the directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify the empty string (""). Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document</p>   |
| Timeout | <p>A number whose value is the maximum time interval within which you allow the specified program to run. Use 0 timeout to allow infinite waiting.</p>  |
| Return  | <p>A numeric that receives the handle of the created process.</p>   |

**Return value:** Number. Returns the handle of the created process.

**Usage:** You can use [RunAndWait](#) for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed. In order to run a document, its extension must be registered. So that if you want to start [MDB](#) files that has [AutoExec](#) macro you must have [MDB](#) file extension registered as a MS Access database application. Use [RunAndWait](#) when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by [command](#). For example, you can call [RunAsUserAndWait](#) first to execute a program that creates huge data file. Then you call [FileZip](#) statement that compresses the produced data file. After that, you can use [MailSendWithAttachment](#) statement to email the archive created by the [FileZip](#).

**See also:**

Run  
RunAsUserAndWait  
RunWithInput  
ProcessGetExitCode

SendKeys

Wait

---

## RunAsUserAndWait

**Description:** Starts specified program or document and enters an efficient wait state until this process finishes or the [timeout](#) interval elapses. In the latter case, the 24x7 Scheduler forcibly terminates the process. The process is started using different Windows NT user login.

**Syntax:** [RunAsUserAndWait command](#), [dir](#), [timeout](#), [user](#), [password](#), [domain](#), [return](#)

---

| Argument                | Description   |
|-------------------------|---|
| <a href="#">command</a> | <p>A string whose value is the full or partial path and filename of the module to execute.</p> <p>The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer.</p> <p>If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:</p> <ol style="list-style-type: none"><li>1 The directory from which the 24x7 Scheduler loaded.</li><li>2 The current directory.</li><li>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.</li><li>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.</li><li>5 The Windows directory. The name of this directory is WINDOWS or WINNT.</li><li>6 The directories that are listed in the PATH environment variable.</li></ol> <p>You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the PATH environment variable as well.</p> <p>Program name can be followed by <b>command line parameters</b>. You can use <b>macro-parameters</b> inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.</p> |
| <a href="#">dir</a>     | <p>A string whose value is the directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify the empty string ("").</p>   |

|                 |   |
|-----------------|---|
|                 | Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document |
| <b>timeout</b>  | A number whose value is the maximum time interval within which you allow the specified program to run. Use 0 timeout to allow infinite waiting.   |
| <b>user</b>     | A string variable that specifies the user name. This is the name of the user account to log on to.  |
| <b>password</b> | A string variable that specifies the password for the user account specified by <b>user</b> .   |
| <b>domain</b>   | A string variable that specifies the domain or server to log on to. If this parameter is ".", Windows NT searches only the local account database for the account specified in <b>user</b> .                        |
| <b>return</b>   | A numeric variable that receives the handle of the created process.   |

---

**Return value:** Number. Returns the handle of the created process.

**Usage:** You can use [RunAsUserAndWait](#) for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed. In order to run a document, its extension must be registered. So that if you want to start [MDB](#) files that has [AutoExec](#) macro you must have [MDB](#) file extension registered as a MS Access database application. Use [RunAsUserAndWait](#) when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by **command**. For example, you can call [RunAsUserAndWait](#) first to execute a program that creates huge data file. Then you call [FileZip](#) statement that compresses the produced data file. After that, you can use [MailSendWithAttachment](#) statement to email the archive created by the [FileZip](#).



#### Important Notes:

- [RunAsUserAndWait](#) statement is **not supported on Windows 95/98** systems.
- Windows NT considers the **user** as logged on until the process (and all child processes) have ended.
- The process created by the [RunAsUserAndWait](#) statement is non-interactive, that is, it runs on a desktop that is not visible and cannot receive user input. Also, the process inherits the environment of the 24x7 Scheduler, rather than the environment associated with the specified **user**.

---

#### See also:

Run  
RunAsUser  
RunAndWait  
ProcessGetExitCode  
Wait

---

## RunWithInput

**Description:** Starts the specified program or document, sends one or more keystrokes to the active window as if typed at the keyboard, and then enters an efficient wait state until the started process finishes or until the **timeout** interval elapses. In the last case, the 24x7 Scheduler forcibly terminates the process.

**Syntax:** `RunWithInput` *command*, *dir*, *init\_time*, *keystroke*, *timeout*, *return*

| Argument               | Description  |
|------------------------|--|
| <code>command</code>   | <p>A string whose value is the full or partial path and filename of the module to execute. The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer.</p> <p>If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence:</p> <ol style="list-style-type: none"> <li>1 The directory from which the 24x7 Scheduler loaded.</li> <li>2 The current directory.</li> <li>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.</li> <li>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.</li> <li>5 The Windows directory. The name of this directory is WINDOWS or WINNT.</li> <li>6 The directories that are listed in the PATH environment variable.</li> </ol> <p>You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in mind that other programs can modify the PATH environment variable as well.</p> <p>Program name can be followed by <b>command line parameters</b>. You can use <b>macro-parameters</b> inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.</p> |
| <code>dir</code>       | <p>A string whose value is the directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify the empty string (""). Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document</p>  |
| <code>init_time</code> | <p>A number whose value is the time in seconds that you want the 24x7 to wait before sending a <code>keystroke</code> to the spawned program, allowing the program to start and initialize.</p>  |
| <code>keystroke</code> | <p>A string whose value is the keystroke you want to send to the started program.</p>  |
| <code>timeout</code>   | <p>A number whose value is the maximum time interval within which you want the specified program to finish. Use 0 timeout to allow infinite waiting.</p>   |

|                        |  |
|------------------------|--|
| <a href="#">return</a> | A numeric that receives the handle of the created process. |
|------------------------|--|

**Return value:** Number.

**Usage:** You can emulate any keystrokes that you might need to further automate the actions of this program. You can use [RunWithInput](#) for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.

In order to run a document, its extension must be registered. So that if you want to start [MDB](#) files that has [AutoExec](#) macro you must have [MDB](#) file extension registered as a MS Access database application.

Use [RunWithInput](#) when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by [command](#). For instance, you can call [RunWithInput](#) first to execute a program that downloads data from database. You send a [keystroke](#) to emulate an operator login to database and the command to save some data in an external file. Then you can call [RunAndWait](#) statement to execute a program that compresses the produced data file. After that, you can use [MailSendWithAttachment](#) statement to email the archive created before.

**See also:**

Run  
RunAndWait  
ProcessGetExitCode  
SendKeys  
Wait

---

## ProcessGetExitCode

**Description:** Obtains exit code of the last process that was ran from the script using [RunAndWait](#) or [RunWithInput](#) statements.

**Syntax:** [ProcessGetExitCode](#) [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A numeric variable that receives the exit code of the last process. |

**Return value:** Number. Returns the exit code the last process.

**Usage:** [ProcessGetExitCode](#) returns either of the following values:

-1 -- if an error occurs (such as program not found or cannot be started)

or

-100 -- if the process was terminated by 24x7 Scheduler because it timed out

or

[process exit code](#).

**See also:**

RunAndWait

```
RunWithInput  
WindowFind  
ProcessGetWindow
```

---

## SendKeys

**Description:** Sends one or more keystrokes to the active window as if typed at the keyboard.

**Syntax:** `SendKeys keystroke`

| Argument               | Description   |
|------------------------|---|
| <code>keystroke</code> | A string whose value is the keystrokes you want to send |

**Return value:** None.

**Usage:** Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter **A**, use `"A"` for string. To represent more than one character, append each additional character to the one preceding it. To represent the letters **A**, **B**, and **C**, use `"ABC"` for string. The left and right parentheses ( `)` have special meanings to `SendKeys`. To specify one of these characters, enclose it within braces ( `{}` ). For example, to specify the plus sign, use `{+}`. To specify brace characters, use `{[}` and `{]}`.

To specify characters that aren't displayed when you press a key, such as **ENTER** or **TAB**, and keys that represent actions rather than characters, use the codes shown below:

| Key           | Code                                |
|---------------|-------------------------------------|
| BACKSPACE     | {BACKSPACE}, {BS}, or {BKSP}        |
| BREAK         | {BREAK}                             |
| CAPS LOCK     | {CAPSLOCK}, {CAPSLOCK}, or {CAP}    |
| DEL or DELETE | {DELETE} or {DEL}                   |
| DOWN ARROW    | {DOWN}                              |
| END           | {END}                               |
| ENTER         | {ENTER} or {CR}, {RETURN}           |
| ESC           | {ESC} or {ESCAPE}                   |
| HOME          | {HOME}                              |
| INS or INSERT | {INSERT} or {INS}                   |
| LEFT ARROW    | {LEFT}                              |
| NUM LOCK      | {NUMLOCK} or {NUM LOCK}             |
| PAGE DOWN     | {PGDN}, {PAGE DOWN}, {PAGEDOWN}     |
| PAGE UP       | {PGUP}, {PAGE UP}, or {PAGE UP}     |
| PRINT SCREEN  | {PRTSC}, {PRINT}, or {PRINT SCREEN} |
| RIGHT ARROW   | {RIGHT}                             |
| SCROLL LOCK   | {SCROLLLOCK} or {SCROLL LOCK}       |

|              |  |
|--------------|--|
| TAB          | {TAB}  |
| UP ARROW     | {UP}   |
| F1           | {F1}   |
| F2           | {F2}   |
| F3           | {F3}   |
| F4           | {F4}   |
| F5           | {F5}   |
| F6           | {F6}   |
| F7           | {F7}   |
| F8           | {F8}   |
| F9           | {F9}   |
| F10          | {F10}  |
| F11          | {F11}  |
| F12          | {F12}  |
| Numpad 0     | {NUMPAD0} or {NUMPAD 0}  |
| Numpad 1     | {NUMPAD1} or {NUMPAD 1}  |
| Numpad 2     | {NUMPAD2} or {NUMPAD 2}  |
| Numpad 3     | {NUMPAD3} or {NUMPAD 3}  |
| Numpad 4     | {NUMPAD4} or {NUMPAD 4}  |
| Numpad 5     | {NUMPAD5} or {NUMPAD 5}  |
| Numpad 6     | {NUMPAD6} or {NUMPAD 6}  |
| Numpad 7     | {NUMPAD7} or {NUMPAD 7}  |
| Numpad 8     | {NUMPAD8} or {NUMPAD 8}  |
| Numpad 9     | {NUMPAD9} or {NUMPAD 9}  |
| Numpad *     | {NUMPAD*} or {NUMPAD *}  |
| Numpad +     | {NUMPAD+} or {NUMPAD +}  |
| Numpad -     | {NUMPAD-} or {NUMPAD -}  |
| Numpad /     | {NUMPAD/} or {NUMPAD /}  |
| Numpad .     | {NUMPAD.} or {NUMPAD .}  |
| Numpad ENTER | {NUMPADENTER}, {NUMPAD ENTER}, {NUMPAD CR}, or {NUMPAD RETURN} |

To specify keys combined with any combination of the [SHIFT](#), [CTRL](#), and [ALT](#) keys, precede the key code with one or more of the following codes:

| Key   | Code    |
|-------|---------|
| SHIFT | {SHIFT} |
| CTRL  | {CTRL}  |
| ALT   | {ALT}   |

To specify that any combination of [SHIFT](#), [CTRL](#), and [ALT](#) should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down [SHIFT](#) while [E](#) and [C](#) are pressed, use "{SHIFT}(EC)". To specify to hold down [SHIFT](#) while [E](#) is pressed, followed by [C](#) without [SHIFT](#), use "{SHIFT}EC" .

To specify that a delay is needed between sending several keys, use the `{WAIT}` code. This will put the 24x7 Scheduler in a 1 second sleep mode. To specify longer delay, use the form `{WAIT number}`. You must put a space between `WAIT` and `number`. For example, `{WAIT 5}` means sleep for 5 seconds.

**Note:**

You can't use `SendKeys` to send keystrokes to an application that is not designed to run in Microsoft Windows.

**See also:**

`RunWithInput`  
`WindowActivate`  
`Wait`

---

## Wait

**Description:** Enters an efficient wait state until the `waitinterval` elapses.

**Syntax:** `Wait waitinterval`

| Argument                  | Description   |
|---------------------------|---|
| <code>Waitinterval</code> | A number whose value is the time interval expressed in seconds during which you want the 24x7 Scheduler to "sleep". |

**Return value:** None.

**Usage:** You can use `Wait` to temporarily pause script execution. For example you may want to pause the script for a few second to allow other processing to finish before continuing the script.

**See also:**

`Run`  
`RunAndWait`

---

## WindowGetProcess

**Description:** Obtains process ID for the specified window.

**Syntax:** `WindowGetProcess handle, return`

| Argument            | Description  |
|---------------------|--|
| <code>handle</code> | A number whose value is the handle of the window for which you want to find its process ID |
| <code>return</code> | A numeric variable that receives the returned value  |

**Return value:** Number. Return process ID for the specified window.

**Usage:** Use [WindowGetProcess](#) to pass returned value to the [ProcessKill](#) statement when you cannot gracefully close that process and want to terminate the process anyway.

**See also:**

[ProcessGetWindow](#)  
[ProcessKill](#)  
[ProcessGetID](#)  
[WindowClose](#)  
[WindowFind](#)  
[WindowGetActive](#)

---

## RAS statements

---

### RASDial

**Description:** Establishes a RAS (remote access service) connection between a RAS client and a RAS server.

**Syntax:** [RASDial](#) [phonebook\\_entry](#), [user\\_name](#), [password](#), [return](#)

| Argument                        | Description  |
|---------------------------------|--|
| <a href="#">Phonebook_entry</a> | A string whose value is the phonebook entry to use to establish the connection.  |
| <a href="#">user_name</a>       | A string whose value is the user's user name. This string is used to authenticate the user's access to the remote access server. |
| <a href="#">Password</a>        | A string whose value is the user's password. This string is used to authenticate the user's access to the remote access server.  |
| <a href="#">Return</a>          | A numeric variable that receives the returned value.   |

**Return value:** Number. The returned value is the RAS connection handle. Use the returned value as a parameter for other RAS statements.

**Usage:** 24x7 Scheduler executes [RASDial](#) statement synchronously meaning that the control is not returned and the next statement is not executed until [RASDial](#) has completed successfully or failed.

**See also:**

[RASHangUp](#)  
[RASGetStatus](#)

## RASGetStatus

**Description:** Retrieves information on the current status of the specified RAS (remote access service) connection.

**Syntax:** `RASGetStatus connection, return`

| Argument   | Description   |
|------------|---|
| Connection | A number whose value is the connection handle returned by <code>RASDial</code> statement. |
| Return     | A string variable that receives the returned value.                                       |

**Return value:** String. Returns information on the current status of the specified remote access connection. You can use this statement to determine the connection status before starting or closing other applications that use RAS connections. The returned status can be one of the following:

- Open Port
- Port Opened
- Connect Device
- Device Connected
- All Devices Connected
- Authenticate
- Auth. Notify
- Auth. Retry
- Auth. Callback
- Auth. Change Password
- Auth. Project
- Auth. Link Speed
- Auth. Ack
- ReAuthenticate
- Authenticated
- Prepare For Callback
- Wait For Modem Reset
- Wait For Callback
- Connected

**See also:**

RASDial  
RASHangUp

---

## RASHangUp

**Description:** Terminates a RAS (remote access service) connection previously established using `RASDial` statement.

**Syntax:** `RASHangUp connection`

| Argument                   | Description  |
|----------------------------|--|
| <a href="#">Connection</a> | A number whose value is the connection handle returned by <a href="#">RASDial</a> statement. |

Return value: None.

**See also:**

[RASDial](#)  
[RASGetStatus](#)

---

## Registry statements

---

### RegistryGetKey

**Description:** Gets a value from the Windows system registry.

**Syntax:** [RegistryGetKey](#) *key*, *valuenam*, *valuetype*, *return*

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">key</a>       | A string whose value names the key in the system registry whose value you want to retrieve. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes |
| <a href="#">valuenam</a>  | A string containing the name of a value in the registry. Each key can have one unnamed value and several named values. For the unnamed value, specify an empty string  |
| <a href="#">valuetype</a> | A string specifying the type of a value in the registry. The following types are supported: <ul style="list-style-type: none"><li>• <a href="#">"STRING"</a> — A null-terminated string</li><li>• <a href="#">"NUMBER"</a> — A 32-bit number</li></ul>           |
| <a href="#">return</a>    | A variable corresponding to the data type of <a href="#">valuetype</a> that receives the returned value  |

**Return value:** String or Number. The result type corresponds to the data type of [valuetype](#).

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- [HKEY\\_CLASSES\\_ROOT](#)
- [HKEY\\_LOCAL\\_MACHINE](#)
- [HKEY\\_USERS](#)
- [HKEY\\_CURRENT\\_USER](#)

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.

[HKEY\\_LOCAL\\_MACHINE\Software\Microsoft\Access\7.0\Options](#)  
[HKEY\\_CURRENT\\_USER\Control Panel\International](#)

**See also:**

[IniFileGetKey](#)  
[IniFileSetKey](#)  
[RegistrySetKey](#)

---

## RegistrySetKey

**Description:** Sets the value for a key and value name in the system registry. If the key or value name does not exist, [RegistrySetKey](#) creates a new key or name and sets its value.

**Syntax:** [RegistrySetKey](#) key, valuename, valuetype, value

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">key</a>       | A string whose value names the key in the system registry whose value you want to set. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes. If key does not exist in the registry, <a href="#">RegistrySetKey</a> creates a new key |
| <a href="#">valuename</a> | A string containing the name of a value in the registry. Each key can have one unnamed value and several named values. For the unnamed value, specify an empty string  |
| <a href="#">valuetype</a> | A string specifying the type of a value in the registry. The following types are supported: <ul style="list-style-type: none"><li>• <a href="#">"STRING"</a> — A null-terminated string</li><li>• <a href="#">"NUMBER"</a> — A 32-bit number</li></ul>   |
| <a href="#">value</a>     | A constant or a variable corresponding to the data type of <a href="#">valuetype</a> containing a value to be set in the registry  |

**Return value:** None.

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- [HKEY\\_CLASSES\\_ROOT](#)
- [HKEY\\_LOCAL\\_MACHINE](#)
- [HKEY\\_USERS](#)
- [HKEY\\_CURRENT\\_USER](#)

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.

[HKEY\\_LOCAL\\_MACHINE\Software\Microsoft\Access\7.0\Options](#)  
[HKEY\\_CURRENT\\_USER\Control Panel\International](#)

**See also:**

IniFileGetKey  
IniFileSetKey  
RegistryGetKey

---

## Service statements

---

### ServiceContinue

**Description:** Resume the execution of a paused Windows NT service

**Syntax:** [ServicePause service](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">service</a> | A string whose value is the name of the paused service that you want to resume. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT platform only. You must have sufficient authority to resume the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The [ServiceGetStatus](#) statement can be used to retrieve the status of a service.

**See also:**

ServicePause  
ServiceStart  
ServiceGetStatus

---

### ServiceGetStatus

**Description:** Retrieves the status of a Windows NT service

**Syntax:** [ServiceGetStatus service](#), [return](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">service</a> | A string whose value is the name of the service whose status you want to retrieve. Service name is case insensitive. |

|                     |  |
|---------------------|--|
| <code>return</code> | A string variable that receives the returned value |
|---------------------|--|

**Return value:** String. The returned value can be one of the following:

| Value              | Meaning  |
|--------------------|--|
| "STOP PENDING"     | The service is stopping.                         |
| "STOPPED"          | The service is not running.                      |
| "START PENDING"    | The service is starting.                         |
| "RUNNING"          | The service is running.                          |
| "CONTINUE PENDING" | The service continue is pending.                 |
| "PAUSE PENDING"    | The service pause is pending.                    |
| "PAUSED"           | The service is paused.                           |
| "ERROR"            | An error occurred while querying service status. |

**Usage:** This statement can be used on Windows NT platform only. You must have sufficient authority to query status of the specified service.

---

## ServicePause

**Description:** Pause the execution of a Windows NT service

**Syntax:** `ServicePause service`

| Argument             | Description   |
|----------------------|---|
| <code>service</code> | A string whose value is the name of the service that you want to suspend. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT platform only. You must have sufficient authority to pause the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The `ServiceGetStatus` statement can be used to retrieve the status of a service.

**See also:**

- ServiceContinue
- ServiceStart
- ServiceGetStatus

## ServiceStart

**Description:** Starts the execution of a Windows NT service

**Syntax:** [ServiceStart service](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">service</a> | A string whose value is the name of the service that you want to start. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT platform only. You must have sufficient authority to start the specified service. You may want to check the service status before trying to start it. An error occurs if the specified service is already running. The [ServiceGetStatus](#) statement can be used to retrieve the status of a service.

**See also:**

- ServicePause
- ServiceStop
- ServiceGetStatus

---

## ServiceStop

**Description:** Stops the execution of a Windows NT service

**Syntax:** [ServiceStop service](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">service</a> | A string whose value is the name of the service that you want to stop. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT platform only. You must have sufficient authority to stop the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The [ServiceGetStatus](#) statement can be used to retrieve the status of a service.

**See also:**

- ServicePause
- ServiceStart
- ServiceGetStatus

---

## String statements

---

### Asc

**Description:** Converts the first character of a string to its ASCII number value.

**Syntax:** `Asc ( string, result )`

| Argument            | Description  |
|---------------------|--|
| <code>string</code> | A string for which you want the ASCII value of the first character |
| <code>result</code> | A numeric variable that receives the returned value                |

**Return value:** Number. Returns the ASCII value of the first character in `string`.

**Usage:** Use `Asc` to test the case of a character or manipulate text and letters. To find out the case of a character, you can check whether its ASCII value is within the appropriate range.

**See also:**

Char

---

### Char

**Description:** Converts a number to a string character.

**Syntax:** `Char ( number, result )`

| Argument            | Description  |
|---------------------|--|
| <code>number</code> | A number you want to convert to a character        |
| <code>result</code> | A string variable that receives the returned value |

**Return value:** String. Returns the string that consists of a single character whose ASCII value is `number`.

**See also:**

Asc

## Concat

**Description:** Appends one string to the end of another strings, in other words concatenates two strings.

**Syntax:** `Concat s1, s2, return`

| Argument            | Description  |
|---------------------|--|
| <code>s1</code>     | The string to which you want to append the string <code>s2</code>      |
| <code>s2</code>     | The string you want to append to the end of the string <code>s1</code> |
| <code>return</code> | A string variable that receives the returned value                     |

**Return value:** String. The concatenation operation returns the string that is union of `s1` and `s2`.

**See also:**

ConcatEx

---

## ConcatEx

**Description:** Appends one or more strings to the end of each other, in other words concatenates them.

**Syntax:** `ConcatEx s1, s2, s3, ..., return`

| Argument            | Description  |
|---------------------|--|
| <code>s1</code>     | The string to which you want to append the string <code>s2</code>                            |
| <code>s2</code>     | The string you want to append to the end of the string <code>s1</code>                       |
| <code>s3</code>     | The string you want to append to the end of the string <code>s1</code> after <code>s2</code> |
| <code>return</code> | A string variable that receives the returned value   |

**Return value:** String. The concatenation operation returns the string that is union of `s1` and `s2` and so on.

**Usage:** Use `ConcatEx` instead of multiple `Concat` statement to concatenate multiple strings into a large one. Note that the JAL script engine automatically handles all datatype conversions so that if any of `ConcatEx` components is not a string it will be automatically converted to it. The number of components in one `ConcatEx` statement cannot exceed 32765.

**See also:**

Concat

---

## Fill

**Description:** Builds a string of the specified length by repeating the specified characters until the result string is long enough.

**Syntax:** `Fill chars, n, return`

| Argument            | Description  |
|---------------------|--|
| <code>chars</code>  | A string whose value will be repeated to fill the return string  |
| <code>n</code>      | A long whose value is the length of the string you want returned |
| <code>return</code> | A string variable that receives the returned value               |

**Return value:** String. Returns a string `n` characters long filled with repetitions of the characters in the argument `chars`. If the argument `chars` has more than `n` characters, the first `n` characters of `chars` are used to fill the return string. If the argument `chars` has fewer than `n` characters, the characters in `chars` are repeated until the return string has `n` characters.

**See also:**

Space

---

## Format

**Description:** Formats data as a string according to a specified format mask. You can convert and format date, datetime, numeric, and time data. You can also apply a format to a string.

**Syntax:** `Format ( data, format , result )`

| Argument            | Description  |
|---------------------|--|
| <code>data</code>   | The data you want returned as a string with the specified formatting. Data can have a date, datetime, numeric, time, or string data type   |
| <code>format</code> | A string of the masks you want to use to format the data. The masks consist of formatting information specific to the data type of data. The format string can consist of more than one mask, depending on the data type of data. Each mask is separated by a semicolon. |
| <code>result</code> | A string variable that receives the returned value   |

**Return value:** String. Returns data in the specified `format` if it succeeds and the empty string ("" ) if the data type of data does not match the type of mask specified or `format` is not a valid mask.

**Usage:**

Formats for date, datetime, string, and time data can include one mask only. Formats for numeric data can have up to three masks. A format with a single mask handles both positive and negative data. If there are additional masks, the first mask is for positive values, and the additional masks are for negative, and zero values. If the format doesn't match the data type, the 24x7 Scheduler will try to apply the mask, which can produce unpredictable results.

**See also:**

Format symbols

String statement

---

## GetToken

**Description:** Obtains first token from the specified string.

**Syntax:** `GetToken s1, s2, return`

| Argument            | Description   |
|---------------------|---|
| <code>s1</code>     | A string that is token separator in the string <code>s1</code> . This could be a single-character string such as space or a multi-character string. |
| <code>s2</code>     | A string from which you want to extract the first available token   |
| <code>return</code> | A string variable that receives the returned value  |

**Return value:** String. The first token from string `s2` separated by `s1`. If no separator is found in the string `s2` then `GetToken` returns the entire string `s2`.

**Usage:** When the first argument `s2` is a variable then `GetToken` also removes the first found token and the following separator from the string `s2` so that it begins with the next token. When there is no more tokens available then `GetToken` returns the empty string ("" ). You can call `GetToken` in a loop to extract all tokens from the string `s2`.

---

## InStr

**Description:** Finds one string within another string.

**Syntax:** `InStr string1, string2, start, return`

| Argument             | Description   |
|----------------------|---|
| <code>string1</code> | The string in which you want to find <code>string2</code> |
| <code>string2</code> | The string you want to find in <code>string1</code>       |
| <code>start</code>   | The number indicating starting position where the search  |

|                        |   |
|------------------------|---|
|                        | will begin in <a href="#">string1</a> .             |
| <a href="#">return</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns a number whose value is the starting position of the first occurrence of [string2](#) in [string1](#) after the position specified in [start](#). If [string2](#) is not found in [string1](#) or if [start](#) is not within [string1](#), [InStr](#) returns 0.

**Usage:** The [InStr](#) function is case sensitive. Alternatively, you can use [Pos](#) statements which syntax and function is the same.

**See also:**

[Pos](#)

---

## Left

**Description:** Obtains a specified number of characters from the beginning of a string.

**Syntax:** [Left](#) [string](#), [n](#), [return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">string</a> | The string containing the characters you want      |
| <a href="#">n</a>      | The number of characters you want                  |
| <a href="#">return</a> | A string variable that receives the returned value |

**Return value:** String. Returns the leftmost [n](#) characters in [string](#) if it succeeds and the empty string ("" ) if an error occurs. If [n](#) is greater than or equal to the length of the [string](#), [Left](#) returns the entire [string](#). It does not add spaces to make the return value's length equal to [n](#).

**See also:**

[Right](#)

[Mid](#)

---

## Length

**Description:** Reports the length of a string.

**Syntax:** [Len](#) [string](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">String</a> | The string for which you want the length            |
| <a href="#">Return</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns a long containing the length of [string](#) if it succeeds and -1 if an error occurs.

---

## Lower

**Description:** Converts all the characters in a string to lowercase.

**Syntax:** [Lower string](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | The string you want to convert to lowercase letters |
| <a href="#">return</a> | A string variable that receives the returned value  |

**Return value:** String. Returns [string](#) with uppercase letters changed to lowercase if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

Upper

---

## LTrim

**Description:** Removes leading spaces from the beginning of a string.

**Syntax:** [LTrim string](#), [return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">string</a> | The string you want returned with leading blanks deleted |
| <a href="#">return</a> | A string variable that receives the returned value       |

**Return value:** String. Returns a copy of [string](#) with leading blanks deleted if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

RTrim  
Trim

## Match

**Description:** Determines whether a string's value contains a particular pattern of characters.

**Syntax:** `Match string, textpattern, return`

| Argument                 | Description  |
|--------------------------|--|
| <code>string</code>      | The string in which you want to look for a pattern of characters |
| <code>textpattern</code> | A string whose value is the text pattern                         |
| <code>return</code>      | A boolean variable that receives the returned value              |

**Return value:** Boolean. Returns TRUE if `string` matches `textpattern` and FALSE if it does not. Match also returns FALSE if either argument has not been assigned a value or the pattern is invalid.

**Usage:** Match enables you to evaluate whether a string contains a general pattern of characters. To find out whether a string contains a specific substring, use the `Pos` statement.

`Textpattern` is similar to a regular expression. It consists of **metacharacters**, which have special meaning, and ordinary characters, which match themselves. You can specify that the string begin or end with one or more characters from a set, or that it contains any characters except those in a set.

A text pattern consists of **metacharacters**, which have special meaning in the match string, and **non-metacharacters**, which match the characters themselves.

**See also:**

metacharacters and sample patterns

---

## Mid

**Description:** Obtains a specified number of characters from a specified position in a string.

**Syntax:** `Mid string, start, length, return`

| Argument            | Description  |
|---------------------|--|
| <code>string</code> | The string from which you want characters returned   |
| <code>start</code>  | A number specifying the position of the first character you want returned. (The position of the first character of the <code>string</code> is 1)   |
| <code>length</code> | A number whose value is the number of characters you want returned. If <code>length</code> is greater than the number of characters to the right of <code>start</code> , Mid returns the remaining characters in the <code>string</code> |
| <code>return</code> | A string variable that receives the returned value   |

**Return value:** String. Returns characters specified in [length](#) of [string](#) starting at character [start](#). If [start](#) is greater than the number of characters in [string](#), the Mid function returns the empty string (""). If [length](#) is greater than the number of characters remaining after the [start](#) character, Mid returns the remaining characters. The return string is not filled with spaces to make it the specified [length](#).

**See also:**

Left  
Right

---

## Number

**Description:** Converts a string to a number.

**Syntax:** [Number](#) ( [string](#), [result](#) )

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | A string you want returned as a number              |
| <a href="#">result</a> | A numeric variable that receives the returned value |

**Return value:** Number. Returns the contents of [string](#) as a number. If [string](#) is not a valid number, Number returns 0.

**See also:**

String  
Format

---

## Pos

**Description:** Finds one string within another string.

**Syntax:** [Pos](#) [string1](#), [string2](#), [start](#), [return](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">string1</a> | The string in which you want to find <a href="#">string2</a>                                     |
| <a href="#">string2</a> | The string you want to find in <a href="#">string1</a>   |
| <a href="#">start</a>   | The number indicating starting position where the search will begin in <a href="#">string1</a> . |
| <a href="#">return</a>  | A numeric variable that receives the returned value  |

**Return value:** Number. Returns a number whose value is the starting position of the first occurrence of [string2](#) in [string1](#) after the position specified in [start](#). If [string2](#) is not found in [string1](#) or if [start](#) is not within [string1](#), [Pos](#) returns 0.

**Usage:** The [Pos](#) function is case sensitive. Alternatively, you can use [InStr](#) statements which syntax and function is the same.

**See also:**

[InStr](#)

---

## Replace

**Description:** Replaces a portion of one string with another.

**Syntax:** [Replace](#) [string1](#), [start](#), [n](#), [string2](#), [return](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">string1</a> | The string in which you want to replace characters with <a href="#">string2</a>  |
| <a href="#">start</a>   | A long whose value is the number of the first character you want replaced. (The first character in the string is number 1)   |
| <a href="#">N</a>       | A number whose value is the number of characters you want to replace   |
| <a href="#">string2</a> | The string that will replace characters in <a href="#">string1</a> . The number of characters in <a href="#">string2</a> can be greater than, equal to, or fewer than the number of characters you are replacing |
| <a href="#">return</a>  | A string variable that receives the returned value   |

**Return value:** String. Returns the string with the characters replaced if it succeeds and the empty string ("" ) if it fails.

**Usage:**

If the [start](#) position is beyond the end of the [string1](#), [Replace](#) appends [string2](#) to [string1](#). If there are fewer characters after the [start](#) position than specified in [n](#), [Replace](#) replaces all the characters to the right of character [start](#). If [n](#) is zero, then in effect [Replace](#) inserts [string2](#) into [string1](#).

**See also:**

[Concat](#)

## Reverse

**Description:** Reverses the order of characters in a string.

**Syntax:** `Reverse string, return`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | A string whose characters you want to reorder so that the last character is first and the first character is last |
| <code>return</code> | A string variable that receives the returned value  |

**Return value:** String. Returns a string with the characters of `string` in reversed order if it succeeds and the empty string ("" ) if an error occurs.

---

## Right

**Description:** Obtains a specified number of characters from the end of a string.

**Syntax:** `Right string, n, return`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | The string containing the characters you want         |
| <code>n</code>      | A number specifying the number of characters you want |
| <code>return</code> | A string variable that receives the returned value    |

**Return value:** String. Returns the rightmost `n` characters in `string` if it succeeds and the empty string ("" ) if an error occurs. If `n` is greater than or equal to the length of the `string`, `Right` returns the entire `string`. It does not add spaces to make the return value's length equal to `n`.

**See also:**

Left  
Mid

---

## RTrim

**Description:** Removes spaces from the end of a string.

**Syntax:** `RTrim string, return`

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | The string you want returned with trailing blanks deleted |
| <a href="#">return</a> | A string variable that receives the returned value        |

**Return value:** String. Returns a copy of [string](#) with trailing blanks deleted if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

[LTrim](#)

[Trim](#)

---

## Space

**Description:** Builds a string of the specified length whose value consists of spaces.

**Syntax:** [Space n](#), [return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">N</a>      | A number whose value is the length of the string you want filled with spaces |
| <a href="#">Return</a> | A string variable that receives the returned value                           |

**Return value:** String. Returns a string filled with [n](#) spaces if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

[Fill](#)

---

## String

**Description:** Converts data to a string. You can convert and format date, datetime, numeric, and time data.

**Syntax:** [String \( data, result \)](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">data</a>   | The data you want returned as a string with the specified formatting. Data can have a date, datetime, numeric, time, or string data type |
| <a href="#">result</a> | A string variable that receives the returned value   |

**Return value:** String. Returns the string based on the [data](#) value.

**Usage:**

For date, datetime, numeric, and time data, the 24x7 Scheduler uses the system's default format for the returned string. You can use the [Format](#) statement to converting data to a string using user-defined format mask.

**See also:**

[Format](#)

---

## Trim

**Description:** Removes leading and trailing spaces from a string.

**Syntax:** [Trim string](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | The <a href="#">string</a> you want returned with leading and trailing spaces deleted |
| <a href="#">return</a> | A string variable that receives the returned value                                    |

**Return value:** String. Returns a copy of [string](#) with all leading and trailing spaces deleted if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

[RTrim](#)

[Ltrim](#)

---

## Upper

**Description:** Converts all the characters in a string to uppercase.

**Syntax:** [Upper string](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | The string you want to convert to uppercase letters |
| <a href="#">return</a> | A string variable that receives the returned value  |

**Return value:** String. Returns [string](#) with lowercase letters changed to uppercase if it succeeds and the empty string ("" ) if an error occurs.

**See also:**

Lower

---

## Telnet statements

---

### TelnetClose

**Description:** Closes active Telnet session.

**Syntax:** [TelnetClose](#)

**Usage:** This statement has no arguments. [TelnetClose](#) closes Telnet session that you previously opened using [TelnetOpen](#) statement.

**See also:**

[TelnetOpen](#)

---

### TelnetOpen

**Description:** Opens new Telnet session.

**Syntax:** [TelnetOpen](#) [host](#), [user](#), [password](#)

| Argument                 | Description  |
|--------------------------|--|
| <a href="#">Host</a>     | A string whose value is the host name of the remote computer (for example, <a href="#">mycompany.com</a> ) or the IP number of the site in ASCII dotted-decimal format (for example, <a href="#">11.0.1.45</a> ) |
| <a href="#">User</a>     | A string whose value is the name of the user to log on   |
| <a href="#">Password</a> | A string whose value is the password to use to log on  |

**Return value:** none.

**Usage:** One job may have one Telnet session open at a time. However, multiple jobs may have multiple Telnet sessions opened simultaneously.

[TelnetOpen](#) statements opens new connection to the remote [host](#) then attempts to intercept the prompts for user name and password. By default it expects the [host](#) to prompt for the user name in the form of "**login:**". After the

prompt for user is received and the name is provided, it expects the [host](#) to prompt for the password in the form of "password:"

If the [host](#) uses different prompts, call [TelnetConfig](#) statement to change the default prompts before you call [TelnetOpen](#) statement.

[Host](#) can be any computer using any Operation System that supports standard Telnet protocol. Telnet protocol is supported by most popular Operation Systems including UNIX, Windows NT, Windows 2000, mainframe computers, etc. **In order to connect to the [host](#) and execute any valid [host](#) commands you do not need 24x7 Scheduler or 24x7 Remote Agent running on the [host](#) computer.**



**Tip:** To troubleshoot connection problems use [TelnetConfig](#) statement to unhide the hidden Telnet terminal window, which displays all messages and errors. After you are done with setup and debugging of your script, comment out or remove the line with [TelnetConfig](#) statement that unhides the terminal.

**See also:**

- [TelnetConfig](#)
- [TelnetClose](#)
- [TelnetSend](#)
- [TelnetReceive](#)

---

## TelnetSend

**Description:** Sends a command or a data to the remote host using active Telnet session.

**Syntax:** [TelnetSend text](#)

| Argument             | Description   |
|----------------------|---|
| <a href="#">Text</a> | A string whose value you want to send as a command or data to the remote host |

**Return value:** none.

**Usage:** [TelnetSend](#) statement simulates an user manually typing some text in the standard Telnet terminal window then pressing the Enter key to submit the entered command or data. [TelnetSend](#) statement automatically adds carriage return (CR) character (ASCII code 13). If the [text](#) is sent as a command and it contains one or more CR characters then every part of the [text](#) separated by the CR will be considered by [host](#) as a separate command.



**Tip:** To troubleshoot various Telnet problems use [TelnetConfig](#) statement to unhide the hidden Telnet terminal window, which displays all messages and errors. After you are done with setup and debugging of your script, comment out or remove the line with [TelnetConfig](#) statement that unhides the terminal.

**See also:**

- [TelnetConfig](#)
- [TelnetOpen](#)
- [TelnetReceive](#)

## TelnetReceive

**Description:** Returns buffered data received from the remote host using active Telnet session.

**Syntax:** [TelnetReceive return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A string variable that receives the returned value. |

**Return value:** String. Returns all data received from the host computer since last [TelnetSend](#) or [TelnetReceive](#) operation.

**Usage:** 24x7 Scheduler stores all text data it receives from the host computer in the internal buffer. Whenever you call [TelnetReceive](#) statement, it simply returns data from the buffer. It is up to you how you interpret or use this data.



**Important Note:** Every call to [TelnetSend](#) or [TelnetReceive](#) statements causes the internal buffer to be reset (cleared).

**See also:**

[TelnetConfig](#)  
[TelnetOpen](#)  
[TelnetSend](#)

---

## TelnetConfig

**Description:** Changes various settings for the active or pending Telnet session.

**Syntax:** [TelnetConfig property, new\\_value](#)

| Argument                  | Description  |
|---------------------------|--|
| <a href="#">Property</a>  | A string whose is the name of the property for the Telnet session that you want to change. The following properties are supported: <ul style="list-style-type: none"><li>• <a href="#">"PROTOCOL"</a> – currently <a href="#">"telnet"</a> protocol is only supported</li><li>• <a href="#">"LOGIN PROMPT"</a> - default value is <a href="#">"login:"</a></li><li>• <a href="#">"PASSWORD PROMPT"</a> - default value is <a href="#">"password:"</a></li><li>• <a href="#">"TIMEOUT"</a> – specify <a href="#">new_value</a> in seconds. Default value is 10 seconds</li><li>• <a href="#">"TERMINAL"</a> – specify either <a href="#">"SHOW"</a> or <a href="#">"HIDE"</a> for the <a href="#">new_value</a></li></ul> |
| <a href="#">New_value</a> | A string whose value is the new value for the <a href="#">property</a> that you want to change.  |

**Return value:** None.

**Usage:** Use [TelnetConfig](#) statement to configure and troubleshoot Telnet sessions. Both parameter names and their values are case insensitive.



**Tips:**

- To unhide the full featured Telnet terminal window use [TelnetConfig](#) "TERMINAL", "SHOW".
- To change the default logon prompt to "user name:" use [TelnetConfig](#) "LOGIN PROMPT", "user name:" before you call [TelnetOpen](#) statement.
- If you are using slow connection to the remote host, increase the timeout value, for example [TelnetConfig](#) "TIMEOUT", "30".

**See also:**

TelnetOpen  
TelnetSend

---

## Window statements

---

### WindowActivate

**Description:** Puts the process that created the specified window into the foreground and activates the window. Keyboard input is directed to the window, and various visual cues are changed for the user.

**Syntax:** [WindowActivate](#) handle

| Argument               | Description   |
|------------------------|---|
| <a href="#">handle</a> | A number whose value is the handle of the window that you want to send in front of other windows and make activate. |

**Return value:** None.

**Usage:** Use [WindowActivate](#) statement before sending keystrokes to desired window.

**See also:**

WindowFind  
WindowGetActive  
SendKeys

## WindowClickButton

**Description:** Searches the specified window for a control having the specified caption and simulates mouse click on the found control.

**Syntax:** [WindowClickButton handle, caption](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">handle</a>  | A number whose value is the handle of the window that owns the control with the specified <a href="#">caption</a> . Controls are buttons, pictures, radio-buttons, check-boxes, etc... |
| <a href="#">caption</a> | A string whose value is the caption of the control that you want to "click" .  |

**Return value:** None.

**Usage:** If the specified window is a parent of other windows, [WindowClickButton](#) automatically searches all child windows. It "clicks" the first control that has the specified [caption](#). Use [WindowFind](#) or any other appropriate Window statement to retrieve [handle](#) of the desired parent window.

**See also:**

[WindowGetActive](#)  
[WindowFind](#)  
[SendKeys](#)

---

## WindowClose

**Description:** Closes the specified window.

**Syntax:** [WindowClose handle](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">handle</a> | A number whose value is the handle of the window that you want to close |

**Return value:** None.

**Usage:** If the specified window is a parent of other windows, [WindowClose](#) automatically closes children before parent. However, there is no guarantee that the specified window will be closed. In some situations when there are some changes pending, the application that owns the specified window may invoke a dialog box, which prompts the user to save changes before exiting. Any processing that the application does on exit will be carried out as usual.



**Notes:**

- Often closing of the main window causes that process to finish. It is highly recommended to use such method instead of using [ProcessKill](#) statement.

- Sending the ALT+F4 keystroke to the active window in most cases causes window closing as well as [WindowClose](#). The CTRL+F4 keystroke closes sheets in multi-document interface systems such as MS Excel.

**See also:**

WindowWaitClose

WindowFind

SendKeys

---

## WindowFind

**Description:** Retrieves the handle of the window whose title matches the specified strings. The specified strings may include the percent (%) sign to indicate a wildcard.

**Syntax:** [WindowFind](#) title, return

| Argument               | Description   |
|------------------------|---|
| <a href="#">title</a>  | A string whose value is the full or partial title of the window that you want to find |
| <a href="#">return</a> | A numeric variable that receives the returned value                                   |

**Return value:** Number. Return a handle of the first found window whose title matches the specified string.

**Usage:** Use search with a percent (%) sign added to the end of the search string when the desired window title varies. This solves the problem with applications that change their titles depending on the document loaded. The percent sign matches any group of characters. For example a search for "[Microsoft Word %](#)" will find an instance of Microsoft Word with any document loaded.

**Note:**

[WindowFind](#) searches invisible windows as well as visible windows

**See also:**

WindowGetActive

WindowGetChild

WindowGetParent

WindowGetProcess

WindowGetTitle

SendKeys

---

## WindowGetActive

**Description:** Obtains the handle of the active window. An active window is the window that receives keyboard input.

**Syntax:** [WindowGetActive](#) return

| Argument               | Description   |
|------------------------|---|
| <a href="#">return</a> | A numeric variable that receives the returned value |

**Return value:** Number. Return a handle of the active window.

**See also:**

WindowFind  
WindowGetProcess  
WindowGetTitle

---

## WindowGetChild

**Description:** Obtains the handle of the first child window for the specified parent window.

**Syntax:** [WindowGetChild](#) [handle](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">handle</a> | A number whose value is the handle of a parent window which owns the window that you want to obtain |
| <a href="#">return</a> | A numeric variable that receives the returned value   |

**Return value:** Number. Return a handle of the child window.

**Usage:** [WindowGetChild](#) returns a handle of the first child window reported by Windows. You can use [WindowGetNext](#) and [WindowGetPrevious](#) statements to obtains handles of other child windows for the same parent window. If the statement succeeds, the return value is a window handle. If no child window exists for the specified window, the return value is [0](#).

**See also:**

WindowFind  
WindowGetTitle  
WindowGetFirst  
WindowGetLast  
WindowGetNext  
WindowGetParent  
WindowGetPrevious

---

## WindowGetFirst

**Description:** Obtains the handle of the first window that has the same type as the specified window.

**Syntax:** [WindowGetFirst](#) handle, return

| Argument               | Description   |
|------------------------|---|
| <a href="#">handle</a> | A number whose value is the handle of the window for which you want to obtain the first window of the same type |
| <a href="#">return</a> | A numeric variable that receives the returned value   |

**Return value:** Number. The retrieved handle identifies the window of the same type that is highest in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is highest in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is highest in the Z order. If the specified window is a child window, the handle identifies the sibling window that is highest in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is [0](#).

**See also:**

- [WindowFind](#)
- [WindowGetTitle](#)
- [WindowGetChild](#)
- [WindowGetLast](#)
- [WindowGetNext](#)
- [WindowGetParent](#)
- [WindowGetPrevious](#)

---

## WindowGetLast

**Description:** Obtains the handle of the last window that has the same type as the specified window.

**Syntax:** [WindowGetLast](#) handle, return

| Argument               | Description  |
|------------------------|--|
| <a href="#">handle</a> | A number whose value is the handle of the window for which you want to obtain the last window of the same type |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. The retrieved handle identifies the window of the same type that is lowest in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is lowest in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is lowest in the Z order. If the specified window is a child window, the handle identifies the sibling window that is lowest in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is [0](#).

**See also:**

- [WindowFind](#)
- [WindowGetTitle](#)
- [WindowGetFirst](#)
- [WindowGetChild](#)
- [WindowGetNext](#)
- [WindowGetParent](#)

---

## WindowGetNext

**Description:** Obtains the handle of the next window that has the same type as the specified window.

**Syntax:** [WindowGetNext handle, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">handle</a> | A number whose value is the handle of the window for which you want to obtain the next window of the same type |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. The retrieved handle identifies the window of the same type that is next in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is next in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is next in the Z order. If the specified window is a child window, the handle identifies the sibling window that is next in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is 0.

**See also:**

WindowFind  
WindowGetTitle  
WindowGetFirst  
WindowGetLast  
WindowGetChild  
WindowGetParent  
WindowGetPrevious

---

## WindowGetParent

**Description:** Obtains the handle of the parent window for the specified child window.

**Syntax:** [WindowGetParent handle, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">handle</a> | A number whose value is the handle of a child window for which you want to obtain it's parent window |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. The retrieved handle identifies the specified window's owner window. If no parent window for the specified window, the return value is 0.

**See also:**

WindowFind  
WindowGetTitle  
WindowGetFirst  
WindowGetLast  
WindowGetNext  
WindowGetPrevious  
WindowGetChild

---

## WindowGetPrevious

**Description:** Obtains the handle of the previous window that has the same type as the specified window.

**Syntax:** [WindowGetPrevious handle, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">handle</a> | A number whose value is the handle of the window for which you want to obtain the previous window of the same type |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. The retrieved handle identifies the window of the same type that is previous in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is previous in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is previous in the Z order. If the specified window is a child window, the handle identifies the sibling window that is previous in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is [0](#).

**See also:**

WindowFind  
WindowGetTitle  
WindowGetFirst  
WindowGetLast  
WindowGetNext  
WindowGetParent  
WindowGetChild

---

## WindowGetProcess

**Description:** Obtains process ID for the specified window.

**Syntax:** [WindowGetProcess handle, return](#)

| Argument               | Description  |
|------------------------|--|
| <a href="#">handle</a> | A number whose value is the handle of the window for which you want to find its process ID |
| <a href="#">return</a> | A numeric variable that receives the returned value  |

**Return value:** Number. Return process ID for the specified window.

**Usage:** Use [WindowGetProcess](#) to pass returned value to the [ProcessKill](#) statement when you cannot gracefully close that process and want to terminate the process anyway.

**See also:**

ProcessGetWindow  
ProcessKill  
ProcessGetID  
WindowClose  
WindowFind  
WindowGetActive

---

## WindowGetTitle

**Description:** Obtains the title of the specified window.

**Syntax:** [WindowGetTitle](#) [handle](#), [return](#)

| Argument               | Description   |
|------------------------|---|
| <a href="#">handle</a> | A number whose value is the handle of the window whose title you want to obtain |
| <a href="#">return</a> | A string variable that receives the returned value                              |

**Return value:** String. Returns window title for the specified window. If the specified window [handle](#) is invalid, the return value is empty string ("").

**See also:**

WindowFind  
WindowGetActive

---

## WindowWaitClose

**Description:** Waits for the specified window to disappear (close). The specified window title may include the percent (%) sign to indicate a wildcard.

**Syntax:** `WindowWaitClose title, timeout`

| Argument             | Description   |
|----------------------|---|
| <code>title</code>   | A string whose value is the full or partial title of the window that you want to find   |
| <code>timeout</code> | A number whose value is the maximum time interval within which you allow the specified window to close. Use <code>0</code> timeout to allow infinite waiting. |

**Return value:** None.

**Usage:** Execution of the script will not continue until the window with the specified `title` text is no longer present or `timeout` occurs. The window `title` may contain the percent (%) symbol at the end to indicate a wildcard. Use it when the desired window title varies. This solves the problem with applications that change their titles depending on the document loaded. The percent sign matches any group of characters. For example a search for "`Microsoft Word %`" will find an instance of Microsoft Word with any document loaded.



**Note:**  
`WindowWaitClose` checks both visible and invisible windows

**See also:**

`WindowWaitOpen`  
`WindowFind`  
`WindowClose`

---

## WindowWaitOpen

**Description:** Waits for the specified window to appear (open). The specified window title may include the percent (%) sign to indicate a wildcard.

**Syntax:** `WindowWaitOpen title, timeout`

| Argument             | Description  |
|----------------------|--|
| <code>title</code>   | A string whose value is the full or partial title of the window that you want to find  |
| <code>timeout</code> | A number whose value is the maximum time interval within which you allow the specified window to open. Use <code>0</code> timeout to allow infinite waiting. |

**Return value:** None.

**Usage:** Execution of the script will not continue until the window with the specified `title` text is found or `timeout` occurs. The window `title` may contain the percent (%) symbol at the end to indicate a wildcard. Use it when the desired window title varies. This solves the problem with applications that change their titles depending on the document loaded. The percent sign matches any group of characters. For example a search for "`Microsoft Word %`" will find an instance of Microsoft Word with any document loaded.

**Note:**

[WindowWaitOpen](#) checks both visible and invisible windows.

**See also:**

[WindowWaitClose](#)

[WindowFind](#)

---

## WindowPostMessage

**Description:** Adds the specified Windows message to the message queue for the specified window.

**Syntax:** [WindowPostMessage](#) [handle](#), [message](#), [word](#), [long](#)

| Argument                | Description   |
|-------------------------|---|
| <a href="#">handle</a>  | A number whose value is the system handle of a window to which you want to post a message                                 |
| <a href="#">message</a> | A number whose value is the system message number of the message you want to send   |
| <a href="#">word</a>    | A number whose value is the word value of the message. If this argument is not used by the message, use <a href="#">0</a> |
| <a href="#">long</a>    | A number whose value is the long value of the message. If this argument is not used by the message, use <a href="#">0</a> |

**Return value:** None.

**Usage:** [WindowPostMessage](#) statement posts the message identified by [message](#) and optionally, [word](#) and [long](#), to the window identified by [handle](#) using Windows SDK function [PostMessage](#). The message is added to the end of the object's message queue. Opposite to [WindowSendMessage](#) statement, [WindowPostMessage](#) is asynchronous; it does not stop execution of the script until the message is processed.

**Note:**

See Windows Software Development Kit (SDK) documentation for complete information on supported messages and parameters

**See also:**

[WindowSendMessage](#)

[Call](#)

## WindowSendMessage

**Description:** Sends the specified Windows message to the specified window so that it is executed immediately.

**Syntax:** `WindowSendMessage handle, message, word, long`

| Argument             | Description  |
|----------------------|--|
| <code>handle</code>  | A number whose value is the system handle of a window to which you want to send a message                              |
| <code>message</code> | A number whose value is the system message number of the message you want to send                                      |
| <code>word</code>    | A number whose value is the word value of the message. If this argument is not used by the message, use <code>0</code> |
| <code>long</code>    | A number whose value is the long value of the message. If this argument is not used by the message, use <code>0</code> |

**Return value:** None.

**Usage:** `WindowSendMessage` statement sends the message identified by `message` and optionally, `word` and `long`, to the window identified by `handle` using Windows SDK function `SendMessage`. The message is sent directly to the object, bypassing the object's message queue. Execution of the script will not continue until the message is processed.



**Note:**

See Windows Software Development Kit (SDK) documentation for complete information on supported messages and parameters

**See also:**

`WindowPostMessage`  
`Call`

---

## Functions in database filter and sort expressions

You can use the following functions in sort and filter expressions. The functions are organized in the following categories.

- Data type checking and conversion functions
- Day, date, and time functions
- Miscellaneous functions
- Numeric functions
- String functions

**See also:**

Operators  
JAL Control-of-Flow Statements  
JAL Statements by Category

---

## Data type checking functions

Asc  
Char  
Date  
DateTime  
Integer  
IsDate  
IsNull  
IsNumber  
IsTime  
Long  
Number  
String  
Time

**See also:**

Functions by category

---

## IsDate

**Description:** Tests whether a string value is a valid date.

**Syntax:** `IsDate ( string )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | A string whose value you want to test to determine whether it is a valid date |

**Return value:** Boolean. Returns TRUE if `datevalue` is a valid date and FALSE if it is not.

---

## IsNull

**Description:** Reports whether the value of a column or expression is NULL.

**Syntax:** `IsNull ( any )`

| Argument         | Description   |
|------------------|---|
| <code>any</code> | A column or expression that you want to test to determine whether its value is NULL |

**Return value:** Boolean. Returns TRUE if `any` is NULL and FALSE if it is not.

**Usage:** Use IsNull to test whether a user-entered value or a value retrieved from the database is NULL.

---

## IsNumber

**Description:** Reports whether the value of a string is a number.

**Syntax:** `IsNumber ( string )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | A string whose value you want to test to determine whether it is a valid number |

**Return value:** Boolean. Returns TRUE if `string` is a valid number and FALSE if it is not.

## IsTime

**Description:** Reports whether the value of a string is a valid time value.

**Syntax:** `IsTime (timevalue)`

| Argument               | Description   |
|------------------------|---|
| <code>timevalue</code> | A string whose value you want to test to determine whether it is a valid time |

**Return value:** Boolean. Returns TRUE if `timevalue` is a valid time and FALSE if it is not.

---

## Date and time functions

---

### Date

**Description:** Converts a string whose value is a valid date to a value of data type date.

**Syntax:** `Date ( string )`

| Argument            | Description  |
|---------------------|--|
| <code>string</code> | A string containing a valid date (such as Jan 1, 1998, or 12-31-99) that you want returned as a date |

**Return value:** Date. Returns the date in `string` as a date. If `string` does not contain a valid date, Date returns NULL.

**Usage:** The value of the string must be a valid date.

Valid dates can include any combination of day (1-31), month (1-12 or the name or abbreviation of a month), and year (two or four digits). Leading zeros are optional for month and day. If the month is a name or an abbreviation, it can come before or after the day; if it is a number, it must be in the month location specified in the Windows control panel. A four-digit number is assumed to be a year.

If the year is two digits, then program chooses the century, as follows. If the year is between 00 and 49, program assumes 20 as the first two digits; if it is between 50 and 99, 24x7 Scheduler assumes 19. If your data includes dates before 1950, such as birth dates, always specify a four-digit year so that 24x7 Scheduler interprets the date as intended.

24x7 Scheduler handles years from 1000 to 3000 inclusive.

An expression has a more limited set of data types than the functions that can be part of the expression. Although the Date function returns a date value, the whole expression is promoted to a DateTime value. Therefore, if your expression consists of a single Date function, it will appear that Date returns the wrong data type. To convert the date without the time, choose an appropriate convert format.

---

## DateTime

**Description:** Combines a date and a time value into a DateTime value.

**Syntax:** `DateTime ( date {, time } )`

| Argument                     | Description   |
|------------------------------|---|
| <code>date</code>            | A valid date (such as Jan 1, 1998, or 12-31-99) or a blob variable whose first value is a date that you want included in the value returned by DateTime   |
| <code>Time (optional)</code> | (optional) A valid time (such as 8AM or 10:25:23:456799) or a blob variable whose first value is a time that you want included in the value returned by DateTime. If you include a time, only the hour portion is required. If you omit the minutes, seconds, or microseconds, they are assumed to be 0s. If you omit AM or PM, the hour is determined according to the 24-hour clock |

**Return value:** DateTime. Returns a DateTime value based on the values in `date` and optionally `time`. If `time` is omitted, DateTime uses 00:00:00.000000 (midnight).

**Usage:** For information on valid dates, see Date function.

---

## Day

**Description:** Obtains the day of the month in a date value.

**Syntax:** `Day (date )`

| Argument          | Description                          |
|-------------------|--------------------------------------|
| <code>date</code> | The date from which you want the day |

**Return value:** Integer. Returns an integer (1-31) representing the day of the month in `date`.

---

## DayName

**Description:** Determines the day of the week in a date value and returns the weekday's name.

**Syntax:** `DayName (date )`

| Argument          | Description                                     |
|-------------------|---|
| <code>date</code> | The date for which you want the name of the day |

**Return value:** String. Returns a string whose value is the name of the weekday (Sunday, Monday, and so on) for `date`.

---

## DayNumber

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** `DayNumber (date )`

| Argument          | Description  |
|-------------------|--|
| <code>date</code> | The date from which you want the number of the day of the week |

**Return value:** Integer. Returns an integer (1-7) representing the day of the week of `date`. Sunday is day 1, Monday is day 2, and so on.

---

## DaysAfter

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** `DaysAfter ( date1, date2 )`

| Argument            | Description  |
|---------------------|--|
| <code>date1</code>  | A date value that is the start date of the interval being measured |
| <code>date2e</code> | A date value that is the end date of the interval                  |

**Return value:** Long. Returns a long containing the number of days [date2](#) occurs after [date1](#). If [date2](#) occurs before [date1](#), DaysAfter returns a negative number.

---

## Hour

**Description:** Obtains the hour in a time value. The hour is based on a 24-hour clock.

**Syntax:** [Hour \( time \)](#)

| Argument             | Description                                 |
|----------------------|---|
| <a href="#">Time</a> | The time value from which you want the hour |

**Return value:** Integer. Returns an integer (00-23) containing the hour portion of [time](#).

---

## Minute

**Description:** Obtains the number of minutes in the minutes portion of a time value.

**Syntax:** [Minute \( time \)](#)

| Argument             | Description                                    |
|----------------------|--|
| <a href="#">Time</a> | The time value from which you want the minutes |

**Return value:** Integer. Returns the minutes portion of [time](#) (00 to 59).

---

## Month

**Description:** Determines the month of a date value.

**Syntax:** [Month \( date \)](#)

| Argument             | Description                            |
|----------------------|--|
| <a href="#">date</a> | The date from which you want the month |

**Return value:** Integer. Returns an integer (1 to 12) whose value is the month portion of [date](#).

---

## Now

**Description:** Obtains the current time based on the system time of the client machine.

**Syntax:** [Now](#) ( )

**Return value:** Time. Returns the current time based on the system time of the client machine.

---

## RelativeDate

**Description:** Obtains the date that occurs a specified number of days after or before another date.

**Syntax:** [RelativeDate](#) ( [date](#), [n](#) )

| Argument             | Description                              |
|----------------------|--|
| <a href="#">date</a> | A date value                             |
| <a href="#">n</a>    | An integer indicating the number of days |

**Return value:** Date. Returns the date that occurs [n](#) days after [date](#) if [n](#) is greater than 0. Returns the date that occurs [n](#) days before [date](#) if [n](#) is less than 0.

---

## RelativeTime

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** [RelativeTime](#) ( [time](#), [n](#) )

| Argument             | Description              |
|----------------------|--------------------------|
| <a href="#">Time</a> | A time value             |
| <a href="#">n</a>    | A long number of seconds |

**Return value:** Time. Returns the time that occurs [n](#) seconds after [time](#) if [n](#) is greater than 0. Returns the time that occurs [n](#) seconds before [time](#) if [n](#) is less than 0. The maximum return value is 23:59:59.

---

## Second

**Description:** Obtains the number of seconds in the seconds portion of a time value.

**Syntax:** `Second ( time )`

| Argument          | Description                                    |
|-------------------|--|
| <code>Time</code> | The time value from which you want the seconds |

**Return value:** Integer. Returns the seconds portion of time (00 to 59).

---

## SecondsAfter

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** `SecondsAfter ( time1, time2 )`

| Argument           | Description  |
|--------------------|--|
| <code>Time1</code> | A time value that is the start time of the interval being measured |
| <code>Time2</code> | A time value that is the end time of the interval                  |

**Return value:** Long. Returns the number of seconds `time2` occurs after `time1`. If `time2` occurs before `time1`, SecondsAfter returns a negative number.

---

## Time

**Description:** Converts a string to a time data type.

**Syntax:** `Time ( string )`

| Argument            | Description  |
|---------------------|--|
| <code>string</code> | A string containing a valid time (such as 8AM or 10:25) that you want returned as a time data type. Only the hour is required; you do not have to include the minutes, |

|  |  |
|--|--|
|  | seconds, or microseconds of the time or AM or PM. The default value for minutes and seconds is 00 and for microseconds is 000000. AM or PM is determined automatically |
|--|--|

**Return value:** Time. Returns the time in [string](#) as a time data type. If [string](#) does not contain a valid time, Time returns 00:00:00.

---

## Today

**Description:** Obtains the system date and time.

**Syntax:** [Today](#) ( )

**Return value:** DateTime. Returns the current system date and time.

---

## Year

**Description:** Determines the year of a date value.

**Syntax:** [Year](#) ( [date](#) )

| Argument             | Description                           |
|----------------------|---------------------------------------|
| <a href="#">date</a> | The date from which you want the year |

**Return value:** Integer. Returns an integer whose value is a four-digit year adapted from the year portion of [date](#) if it succeeds and 1900 if an error occurs.

If the year is two digits, then program chooses the century, as follows. If the year is between 00 to 49, program assumes 20 as the first two digits; if it is between 50 and 99, program assumes 19.

**Usage:** Obtains the year portion of date. 24x7 Scheduler handles years from 1000 to 3000 inclusive. If your data includes dates before 1950, such as birth dates, always specify a 4-digit year so that Year function (and other functions) interprets the date as intended.

---

## Miscellaneous functions

Case  
If

ProfileInt  
ProfileString  
RGB

---

## Case

**Description:** Tests the values of a column or expression and returns values based on the results of the test.

**Syntax:**

Case ( column WHEN value1 THEN result1 { WHEN value2 THEN result2 { ... } { ELSE resultelse } )

| Argument        | Description  |
|-----------------|--|
| Column          | The column or expression whose values you want to test. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. Column is compared to each <a href="#">valuen</a>   |
| WHEN            | Introduces a value-result pair. At least one WHEN is required  |
| Valuen          | One or more values that you want to compare to values of column. A value can be: <ul style="list-style-type: none"><li>• A single value</li><li>• A list of values separated by commas (for example, 2, 4, 6, 8)</li><li>• A TO clause (for example, 1 TO 20)</li><li>• IS followed by a relational operator and comparison value (for example, IS&gt;5)</li><li>• Any combination of the above with an implied OR between expressions (for example, 1,3,5,7,9,27 TO 33, IS&gt;42)</li></ul> |
| THEN            | Introduces the result to be returned when column matches the corresponding <a href="#">valuen</a>  |
| Resultn         | An expression whose value is returned by Case for the corresponding <a href="#">valuen</a> . All <a href="#">resultn</a> must have the same data type  |
| ELSE (optional) | (optional) Specifies that for any values of column that don't match the values of <a href="#">valuen</a> already specified, Case returns <a href="#">resultelse</a>  |
| Resultelse      | An expression whose value is returned by Case when the value of column doesn't match any WHEN valuen expression  |

**Return value:** The data type of [resultn](#). Returns the result you specify in [resultn](#).

**Usage:** If more than one WHEN clause matches column, Case returns the result of the first matching one.



**Note:**

Function Case is similar to the ORACLE's Decode function.

## GetRow

**Description:** Reports the number of a row in which processing is performed.

**Syntax:** `GetRow ( )`

**Return value:** Long. Returns the number of a row if it succeeds, 0 if no data has been retrieved or added, and -1 if an error occurs.

---

## If

**Description:** Evaluates a condition and returns a value based on that condition.

**Syntax:** `If ( boolean, truevalue, falsevalue )`

| Argument                | Description   |
|-------------------------|---|
| <code>boolean</code>    | A boolean expression that evaluates to TRUE or FALSE  |
| <code>Truevalue</code>  | A string containing the value you want returned if the <code>boolean</code> expression is TRUE  |
| <code>Falsevalue</code> | A string containing the value you want returned if the <code>boolean</code> expression is FALSE |

**Return value:** The data type of `truevalue` or `falsevalue`. Returns `truevalue` if `boolean` is TRUE and `falsevalue` if it is FALSE. Returns NULL if an error occurs.

---

## IsRowModified

**Description:** Reports whether the row has been modified.

**Syntax:** `IsRowModified ( )`

**Return value:** Boolean. Returns TRUE if the row has been modified and FALSE if it has not.

**Example:** To filter out only modified rows, set filter expression as following:  
`IsRowModified ( )`

## IsRowNew

**Description:** Reports whether the row has been newly inserted after last save or retrieve.

**Syntax:** `IsRowNew ( )`

**Return value:** Returns TRUE if the row is new and FALSE if it was retrieved from the database.

**Example:** To filter out only newly inserted rows, set filter expression as following:  
`IsRowNew ( )`

**See also:**

Functions by category

---

## ProfileInt

**Description:** Obtains the integer value of a setting in the specified profile file.

**Syntax:** `ProfileInt ( filename, section, key, default )`

| Argument              | Description  |
|-----------------------|--|
| <code>Filename</code> | A string whose value is the name of the profile file. If you do not specify a full path, ProfileInt uses the operating system's standard file search order to find the file  |
| <code>section</code>  | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive                            |
| <code>Key</code>      | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in <code>section</code> . <code>Section</code> is not case-sensitive |
| <code>default</code>  | An integer value that ProfileInt will return if <code>filename</code> is not found, if <code>section</code> or <code>key</code> does not exist in <code>filename</code> , or if the value of key cannot be converted to an integer     |

**Return value:** Integer. Returns default if `filename` is not found, `section` is not found in `filename`, or `key` is not found in `section`, or the value of `key` is not an integer. Returns -1 if an error occurs.

**Usage:** Use ProfileInt or ProfileString to get configuration settings from a profile file.

## ProfileString

**Description:** Obtains the string value of a setting in the specified profile file.

**Syntax:** `ProfileInt ( filename, section, key, default )`

| Argument | Description  |
|----------|--|
| Filename | A string whose value is the name of the profile file. If you do not specify a full path, ProfileString uses the operating system's standard file search order to find the file   |
| Section  | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive                                  |
| Key      | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in <a href="#">section</a> . <a href="#">Section</a> is not case-sensitive |
| default  | An string value that ProfileString will return if <a href="#">filename</a> is not found, if <a href="#">section</a> or <a href="#">key</a> does not exist in <a href="#">filename</a>  |

**Return value:** String, with a maximum length of 4096 characters. Returns the string from [key](#) within [section](#) within [filename](#). If [filename](#) is not found, [section](#) is not found in [filename](#), or [key](#) is not found in [section](#), ProfileString returns [default](#). If an error occurs, it returns the empty string ("").

**Usage:** Use ProfileInt or ProfileString to get configuration settings from a profile file.

---

## RGB

**Description:** Calculates the long value that represents the color specified by numeric values for the red, green, and blue components of the color.

**Syntax:** `RGB ( red, green, blue )`

| Argument | Description   |
|----------|---|
| Red      | The integer value of the red component of the desired color   |
| green    | The integer value of the green component of the desired color |
| blue     | The integer value of the blue component of the desired color  |

**Return value:** Long. Returns the long that represents the color created by combining the values specified in [red](#), [green](#), and [blue](#). If an error occurs, RGB returns NULL.

**Usage:** The formula for combining the colors is:

Red + (256 \* Green) + (65536 \* Blue)

Use RGB to obtain the long value required to set the color for text and drawing objects. You can also set an object's color to the long value that represents the color. The RGB function provides an easy way to calculate that value. The value of a component color is an integer between 0 and 255

---

## Numeric functions

Abs  
Ceiling  
Cos  
Exp  
Fact  
Int  
Log  
LogTen  
Mod  
Pi  
Rand  
Round  
Sign  
Sin  
Sqrt  
Tan  
Truncate

---

### Abs

**Description:** Calculates the absolute value of a number.

**Syntax:** [Abs](#) ( [n](#) )

| Argument          | Description                                      |
|-------------------|--|
| <a href="#">n</a> | The number for which you want the absolute value |

**Return value:** The data type of [n](#). Returns the absolute value of [n](#).

---

## Ceiling

**Description:** Determines the smallest whole number that is greater than or equal to a specified limit.

**Syntax:** `Ceiling ( n )`

| Argument       | Description   |
|----------------|---|
| <code>N</code> | The number for which you want the smallest whole number that is greater than or equal to it |

**Return value:** The data type of `n`. Returns the smallest whole number that is greater than or equal to `n`.

---

## Cos

**Description:** Calculates the cosine of an angle.

**Syntax:** `Cos ( n )`

| Argument       | Description  |
|----------------|--|
| <code>n</code> | The angle (in radians) for which you want the cosine |

**Return value:** Double. Returns the cosine of `n`.

---

## Exp

**Description:** Raises e to the specified power.

**Syntax:** `Exp ( n )`

| Argument       | Description                                      |
|----------------|--|
| <code>n</code> | The power to which you want to raise e (2.71828) |

**Return value:** Double. Returns e raised to the power `n`.

---

## Fact

**Description:** Determines the factorial of a number.

**Syntax:** `Fact ( n )`

| Argument       | Description                                 |
|----------------|---|
| <code>n</code> | The number for which you want the factorial |

**Return value:** Double. Returns the factorial of `n`.

---

## Int

**Description:** Determines the largest whole number less than or equal to a number.

**Syntax:** `Int ( n )`

| Argument       | Description   |
|----------------|---|
| <code>n</code> | The number for which you want the largest whole number that is less than or equal to it |

**Return value:** The data type of `n`. Returns the largest whole number less than or equal to `n`.

---

## Integer

**Description:** Converts the value of a string to an integer.

**Syntax:** `Integer ( string )`

| Argument            | Description                                |
|---------------------|--|
| <code>string</code> | The string you want returned as an integer |

**Return value:** Integer. Returns the contents of `string` as an integer if it succeeds and 0 if `string` is not a number.

## Log

**Description:** Determines the natural logarithm of a number.

**Syntax:** `Log ( n )`

| Argument       | Description  |
|----------------|--|
| <code>n</code> | The number for which you want the natural logarithm (base <code>e</code> ). The value of <code>n</code> must be greater than 0 |

**Return value:** Double. Returns the natural logarithm of `n`. An execution error occurs if `n` is negative or zero.

**Inverse:** The inverse of the Log function is the Exp function.

---

## LogTen

**Description:** Determines the base 10 logarithm of a number.

**Syntax:** `LogTen ( n )`

| Argument       | Description   |
|----------------|---|
| <code>n</code> | The number for which you want the base 10 logarithm. The value of <code>n</code> must not be negative |

**Return value:** Double. Returns the decimal log of `n`.

**Obtaining a number:** The expression `10^n` is the inverse for `LogTen(n)`. To obtain `n` given number (`nbr = LogTen(n)`), use `n = 10^r`.

---

## Long

**Description:** Converts the value of a string to a long.

**Syntax:** `Long ( string )`

| Argument            | Description                            |
|---------------------|--|
| <code>string</code> | The string you want returned as a long |

**Return value:** Long. Returns the contents of [string](#) as a long if it succeeds and 0 if [string](#) is not a valid number.

---

## Mod

**Description:** Obtains the remainder (modulus) of a division operation.

**Syntax:** [Mod](#) ( [x](#), [y](#) )

| Argument          | Description  |
|-------------------|--|
| <a href="#">X</a> | The number you want to divide by <a href="#">y</a>   |
| <a href="#">Y</a> | The number you want to divide into <a href="#">x</a> |

**Return value:** The data type of [x](#) or [y](#), whichever data type is more precise.

---

## Number

**Description:** Converts a string to a number.

**Syntax:** [Number](#) ( [string](#) )

| Argument               | Description                              |
|------------------------|--|
| <a href="#">string</a> | The string you want returned as a number |

**Return value:** A numeric data type. Returns the contents of [string](#) as a number. If [string](#) is not a valid number, Number returns 0.

---

## Pi

**Description:** Multiplies pi by a specified number.

**Syntax:** [Pi](#) ( [n](#) )

| Argument          | Description   |
|-------------------|---|
| <a href="#">n</a> | The number you want to multiply by <a href="#">pi</a><br>(3.14159265358979323...) |

**Return value:** Double. Returns the result of multiplying *n* by *pi* if it succeeds and -1 if an error occurs.

**Usage:** Use *Pi* to convert angles to and from radians.

---

## Rand

**Description:** Obtains a random whole number between 1 and a specified upper limit.

**Syntax:** *Rand ( n )*

| Argument | Description   |
|----------|---|
| <i>n</i> | The upper limit of the range of random numbers you want returned. The lower limit is always 1. The upper limit cannot exceed 32,767 |

**Return value:** A numeric data type, the data type of *n*. Returns a random whole number between 1 and *n*.

**Usage:** The sequence of numbers generated by repeated calls to the Rand function is a computer-generated pseudo-random sequence.

---

## Round

**Description:** Rounds a number to the specified number of decimal places.

**Syntax:** *Round ( x, n )*

| Argument | Description  |
|----------|--|
| <i>x</i> | The number you want to round                                     |
| <i>n</i> | The number of decimal places to which you want to round <i>x</i> |

**Return value:** Decimal. If *n* is positive, returns *x* rounded to the specified number of decimal places. If *n* is negative, returns *x* rounded to *(-n+1)* places before the decimal point. Returns -1 if it fails.

---

## Sign

**Description:** Determines the sign of a number. Sign reports whether a number is negative, zero, or positive.

**Syntax:** `Sign ( n )`

| Argument       | Description   |
|----------------|---|
| <code>n</code> | The number for which you want to determine the sign |

**Return value:** Integer. Returns a number (-1, 0, or 1) indicating the sign of `n`.

---

## Sin

**Description:** Calculates the sine of an angle.

**Syntax:** `Sin ( n )`

| Argument       | Description  |
|----------------|--|
| <code>n</code> | The angle (in radians) for which you want the sine |

**Return value:** Double. Returns the sine of `n`.

---

## Sqrt

**Description:** Calculates the square root of a number.

**Syntax:** `Sqrt ( n )`

| Argument       | Description                                   |
|----------------|---|
| <code>n</code> | The number for which you want the square root |

**Return value:** Double. Returns the square root of `n`.

**Usage:** `Sqrt(n)` is the same as `n^.5`.

Taking the square root of a negative number causes an execution error.

## Tan

**Description:** Calculates the tangent of an angle.

**Syntax:** Tan ( n )

| Argument | Description   |
|----------|---|
| n        | The angle (in radians) for which you want the tangent |

**Return value:** Double. Returns the tangent of n if it succeeds and -1 if an error occurs.

---

## Truncate

**Description:** Truncates a number to the specified number of decimal places.

**Syntax:** Truncate ( x, n )

| Argument | Description  |
|----------|--|
| x        | The number you want to truncate                              |
| n        | The number of decimal places to which you want to truncate x |

**Return value:** The data type of n. If n is positive, returns x truncated to the specified number of decimal places. If n is negative, returns x truncated to (-n+1) places before the decimal point. Returns -1 if it fails.

---

## String functions

Fill  
Left  
LeftTrim  
Len  
Lower  
Match  
Mid  
Pos  
Replace  
Right  
RightTrim  
Space

String  
Trim  
Upper  
WordCap

---

## Asc

**Description:** Converts the first character of a string to its ASCII integer value.

**Syntax:** `Asc ( string )`

| Argument | Description  |
|----------|--|
| String   | The string for which you want the ASCII value of the first character |

**Return value:** Integer. Returns the ASCII value of the first character in `string`.

**Usage:** Use Asc to test the case of a character or manipulate text and letters. To find out the case of a character, you can check whether its ASCII value is within the appropriate range.

---

## Char

**Description:** Converts an integer to a character.

**Syntax:** `Char ( n )`

| Argument | Description                                    |
|----------|--|
| n        | The integer you want to convert to a character |

**Return value:** String. Returns the character whose ASCII value is `n`.

---

## Fill

**Description:** Builds a string of the specified length by repeating the specified characters until the result string is long enough.

**Syntax:** `Fill ( chars, n )`

| Argument              | Description  |
|-----------------------|--|
| <a href="#">Chars</a> | A string whose value will be repeated to fill the return string                    |
| <a href="#">n</a>     | A long whose value is the length of the string you want returned (from 0 to 60000) |

**Return value:** String. Returns a string [n](#) characters long filled with repetitions of the characters in the argument [chars](#). If the argument [chars](#) has more than [n](#) characters, the first [n](#) characters of [chars](#) are used to fill the return string. If the argument [chars](#) has fewer than [n](#) characters, the characters in [chars](#) are repeated until the return string has [n](#) characters.

---

## Left

**Description:** Obtains a specified number of characters from the beginning of a string.

**Syntax:** [Left](#) ( [string](#), [n](#) )

| Argument               | Description   |
|------------------------|---|
| <a href="#">string</a> | The string containing the characters you want       |
| <a href="#">n</a>      | A long specifying the number of characters you want |

**Return value:** String. Returns the leftmost [n](#) characters in [string](#) if it succeeds and the empty string ("" ) if an error occurs. If [n](#) is greater than or equal to the length of the [string](#), [Left](#) returns the entire [string](#). It does not add spaces to make the return value's length equal to [n](#).

---

## LeftTrim

**Description:** Removes leading spaces from the beginning of a string.

**Syntax:** [LeftTrim](#) ( [string](#) )

| Argument               | Description  |
|------------------------|--|
| <a href="#">string</a> | The string you want returned with leading blanks deleted |

**Return value:** String. Returns a copy of [string](#) with leading blanks deleted if it succeeds and the empty string ("" ) if an error occurs.

---

## Len

**Description:** Reports the length of a string.

**Syntax:** `Len ( string )`

| Argument            | Description                              |
|---------------------|--|
| <code>string</code> | The string for which you want the length |

**Return value:** Long. Returns a long containing the length of string if it succeeds and -1 if an error occurs.

---

## Lower

**Description:** Converts all the characters in a string to lowercase.

**Syntax:** `Lower ( string )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | The string you want to convert to lowercase letters |

**Return value:** String. Returns `string` with uppercase letters changed to lowercase if it succeeds and the empty string ("" ) if an error occurs.

---

## Match

**Description:** Determines whether a string's value contains a particular pattern of characters.

**Syntax:** `Match ( string, textpattern )`

| Argument                 | Description  |
|--------------------------|--|
| <code>string</code>      | The string in which you want to look for a pattern of characters |
| <code>Textpattern</code> | A string whose value is the text pattern                         |

**Return value:** Boolean. Returns TRUE if `string` matches `textpattern` and FALSE if it does not. Match also returns FALSE if either argument has not been assigned a value or the pattern is invalid.

**Usage:** Match enables you to evaluate whether a string contains a general pattern of characters. To find out whether a string contains a specific substring, use the [Pos](#) function.

[Textpattern](#) is similar to a regular expression. It consists of metacharacters, which have special meaning, and ordinary characters, which match themselves. You can specify that the string begin or end with one or more characters from a set, or that it contains any characters except those in a set.

A text pattern consists of **metacharacters**, which have special meaning in the match string, and **non-metacharacters**, which match the characters themselves.

**See also:**

Metacharacters and sample patterns

## Metacharacters

The following tables explain the meaning and use of the metacharacters:

| Metacharacter   | Meaning   | Example   |
|---|---|---|
| Caret (^)   | Matches the beginning of a string   | ^C matches C at the beginning of a string   |
| Dollar sign (\$)  | Matches the end of a string   | s\$ matches s at the end of a string  |
| Period (.)  | Matches any character   | . . . matches three consecutive characters  |
| Backslash (\)   | Removes the following metacharacter's special characteristics so that it matches itself | \\$ matches \$  |
| Character class (a group of characters enclosed in square brackets ([ ]))     | Matches any of the enclosed characters  | [AEIOU] matches A, E, I, O, or U<br>you can use hyphens to abbreviate ranges of characters in a character class. For example, [A-Za-z] matches any letter |
| Complemented character class (first character inside the brackets is a caret) | Matches any character not in the group following the caret                              | [^0-9] matches any character except a digit, and [^A-Za-z] matches any character except a letter  |

The metacharacters asterisk (\*), plus (+), and question mark (?) are unary operators that are used to specify repetitions in a regular expression:

| Metacharacter     | Meaning                            | Example   |
|-------------------|------------------------------------|---|
| * (asterisk)      | Indicates zero or more occurrences | A* matches zero or more As (no As, A, AA, AAA, and so on) |
| + (plus)          | Indicates one or more occurrences  | A+ matches one A or more than one A (A, AAA, and so on)   |
| ? (question mark) | Indicates zero or one occurrence   | A? matches an empty string ("" ) or A                     |

## Sample patterns

The following table shows various text patterns and sample text that matches each pattern:

| This pattern         | Matches  |
|----------------------|--|
| AB                   | Any string that contains AB; for example, ABA, DEABC, graphAB_one  |
| B*                   | Any string that contains 0 or more Bs; for example, AC, B, BB, BBB, ABBBC, and so on                               |
| AB*C                 | Any string containing the pattern AC or ABC or ABBC, and so on (0 or more Bs)                                      |
| AB+C                 | Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 or more Bs)                                   |
| ABB*C                | Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 B plus 0 or more Bs)                          |
| ^AB                  | Any string starting with AB  |
| AB?C                 | Any string containing the pattern AC or ABC (0 or 1 B)   |
| ^[ABC]               | Any string starting with A, B, or C  |
| [^ABC]               | A string containing any characters other than A, B, or C   |
| ^[^abc]              | A string that begins with any character except a, b, or c  |
| ^[^a-z]\$            | Any single-character string that is not a lowercase letter (^ and \$ indicate the beginning and end of the string) |
| [A-Z]+               | Any string with one or more uppercase letters  |
| ^[0-9]+\$            | Any string consisting only of digits   |
| ^[0-9][0-9][0-9]\$   | Any string consisting of exactly three digits  |
| ^([0-9][0-9][0-9])\$ | Any consisting of exactly three digits enclosed in parentheses   |

### See also:

Match function

---

## Mid

**Description:** Obtains a specified number of characters from a specified position in a string.

**Syntax:** `Mid ( string, start {, length } )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | The string from which you want characters returned        |
| <code>start</code>  | A long specifying the position of the first character you |

|                                   |   |
|-----------------------------------|---|
|                                   | want returned. (The position of the first character of the <a href="#">string</a> is 1)   |
| <a href="#">Length (optional)</a> | A long whose value is the number of characters you want returned. If you do not enter <a href="#">length</a> or if <a href="#">length</a> is greater than the number of characters to the right of <a href="#">start</a> , Mid returns the remaining characters in the <a href="#">string</a> |

**Return value:** String. Returns characters specified in [length](#) of [string](#) starting at character [start](#). If [start](#) is greater than the number of characters in [string](#), the Mid function returns the empty string (""). If [length](#) is greater than the number of characters remaining after the [start](#) character, Mid returns the remaining characters. The return string is not filled with spaces to make it the specified [length](#).

---

## Pos

**Description:** Finds one string within another string.

**Syntax:** [Pos \( string1, string2 {, start } \)](#)

| Argument                         | Description   |
|----------------------------------|---|
| <a href="#">string1</a>          | The string in which you want to find <a href="#">string2</a>                                |
| <a href="#">string2</a>          | The string you want to find in <a href="#">string1</a>                                      |
| <a href="#">start (optional)</a> | A long indicating where the search will begin in <a href="#">string1</a> . The default is 1 |

**Return value:** Long. Returns a long whose value is the starting position of the first occurrence of [string2](#) in [string1](#) after the position specified in [start](#). If [string2](#) is not found in [string1](#) or if [start](#) is not within [string1](#), Pos returns 0.

**Usage:** The Pos function is case sensitive.

---

## Replace

**Description:** Replaces a portion of one string with another.

**Syntax:** [Replace \( string1, start, n, string2 \)](#)

| Argument                | Description  |
|-------------------------|--|
| <a href="#">string1</a> | The string in which you want to replace characters with <a href="#">string2</a>  |
| <a href="#">start</a>   | A long whose value is the number of the first character you want replaced. (The first character in the string is number 1) |

|                |  |
|----------------|--|
| <b>n</b>       | A long whose value is the number of characters you want to replace   |
| <b>string2</b> | The string that will replace characters in <b>string1</b> . The number of characters in <b>string2</b> can be greater than, equal to, or fewer than the number of characters you are replacing |

**Return value:** String. Returns the string with the characters replaced if it succeeds and the empty string ("" ) if it fails.

**Usage:** If the **start** position is beyond the end of the **string1**, Replace appends **string2** to **string1**. If there are fewer characters after the **start** position than specified in **n**, Replace replaces all the characters to the right of character **start**.

If **n** is zero, then in effect Replace inserts **string2** into **string1**.

---

## Right

**Description:** Obtains a specified number of characters from the end of a string.

**Syntax:** **Right** ( **string**, **n** )

| Argument      | Description   |
|---------------|---|
| <b>string</b> | The string containing the characters you want       |
| <b>n</b>      | A long specifying the number of characters you want |

**Return value:** String. Returns the rightmost **n** characters in **string** if it succeeds and the empty string ("" ) if an error occurs. If **n** is greater than or equal to the length of the **string**, Right returns the entire **string**. It does not add spaces to make the return value's length equal to **n**.

---

## RightTrim

**Description:** Removes spaces from the end of a string.

**Syntax:** **RightTrim** ( **string** )

| Argument      | Description   |
|---------------|---|
| <b>string</b> | The string you want returned with trailing blanks deleted |

**Return value:** String. Returns a copy of **string** with trailing blanks deleted if it succeeds and the empty string ("" ) if an error occurs.

---

## Space

**Description:** Builds a string of the specified length whose value consists of spaces.

**Syntax:** `Space ( n )`

| Argument       | Description  |
|----------------|--|
| <code>n</code> | A long whose value is the length of the string you want filled with spaces |

**Return value:** String. Returns a string filled with `n` spaces if it succeeds and the empty string ("" ) if an error occurs.

---

## String

**Description:** Formats data as a string according to a specified format mask. You can convert and format date, DateTime, numeric, and time data. You can also apply a format to a string.

**Syntax:** `String ( data {, format } )`

| Argument                       | Description  |
|--------------------------------|--|
| <code>Data</code>              | The data you want returned as a string with the specified formatting. Data can have a date, DateTime, numeric, time, or string data type   |
| <code>Format (optional)</code> | A string of the masks you want to use to format the data. The masks consist of formatting information specific to the data type of data. If data is type string, format is required. The format string can consist of more than one mask, depending on the data type of data. Each mask is separated by a semicolon. See Usage for details on each data type |

**Return value:** String. Returns data in the specified `format` if it succeeds and the empty string ("" ) if the data type of data does not match the type of mask specified or `format` is not a valid mask.

**Usage:** For date, DateTime, numeric, and time data, program uses the system's default format for the returned string if you don't specify a format. For numeric data, the default format is the [General] format. For string data, a format mask is required. (Otherwise, the function would have nothing to do.) The format can consist of one or more masks:

- Formats for date, DateTime, string, and time data can include one or two masks. The first mask is the format for the data; the second mask is the format for a null value.
- Formats for numeric data can have up to four masks. A format with a single mask handles both positive and negative data. If there are additional masks, the first mask is for positive values, and the additional masks are for negative, zero, and NULL values.

If the format doesn't match the data type, 24x7 Scheduler will try to apply the mask, which can produce unpredictable results.

**See also:**

Format symbols

## Format Symbols

The following table lists the format symbols that can be used in a format string:

| Format Symbol        | Description  |
|----------------------|--|
| [General]            | Outputs the number in General format.  |
| 0                    | Digit placeholder. If the number contains fewer digits than the format contains placeholders, the number is padded with 0's. If there are more digits to the right of the decimal than there are placeholders, the decimal portion is rounded to the number of places specified by the placeholders. If there are more digits to the left of the decimal than there are placeholders, the extra digits are retained.   |
| #                    | Digit placeholder. This placeholder functions the same as the 0 placeholder except the number is not padded with 0's if the number contains fewer digits than the format contains placeholders.  |
| ?                    | Digit placeholder. This placeholder functions the same as the 0 placeholder except that spaces are used to pad the digits.   |
| . (period)           | Decimal point. Determines how many digits (0's or #'s) are output on either side of the decimal point. If the format contains only #'s left of the decimal point, numbers less than 1 begin with a decimal point. If the format contains 0's left of the decimal point, numbers less than 1 begin with a 0 left of the decimal point.  |
| %                    | Outputs the number as a percentage. The number is multiplied by 100 and the % character is appended.   |
| , (comma)            | Thousands separator. If the format contains commas separated by #'s or 0's, the number is output with commas separating thousands. A comma following a placeholder scales the number by a thousand. For Example, the format 0, scales the number by 1000 (e.g., 10,000 would be converted as 10).  |
| E- E+ e- e+          | Outputs the number as scientific notation. If the format contains a scientific notation symbol to the left of a 0 or # placeholder, the number is output in scientific notation and an E or an e is added. The number of 0 and # placeholders to the right of the decimal determines the number of digits in the exponent. E- and e- place a minus sign by negative exponents. E+ and e+ place a minus sign by negative exponents and a plus sign by positive exponents. |
| \$ - + / ( ) : space | Outputs that character. To output a character other than those listed, precede the character with a back slash (\) or enclose the character in double quotation marks ("" ). You can also use the slash (/) for fraction formats.  |
| \                    | Outputs the next character. The backslash is not output. You can also output a character or string of characters by surrounding the characters with double quotation marks ("" ).  |
| * (asterisk)         | Repeats the next character until the width of the column is filled. You cannot have more than one asterisk in each format section.   |
| [ ]                  | Outputs hours greater than 24, or minutes or seconds greater than 60. Place the brackets around the leftmost part of the time code; for Example, [h]:mm:ss would allow the output of hours greater than 24.  |
| "text"               | Outputs the text inside the quotation marks.   |

|                              |  |
|------------------------------|--|
| @                            | Text placeholder. Text replaces the @ format character.  |
| m                            | Month number. Outputs the month as digits without leading zeros (e.g., 1-12). Can also represent minutes when used with h or hh formats.   |
| mm                           | Month number. Outputs the month as digits with leading zeros (e.g., 01-12). Can also represent minutes when used with the h or hh formats.   |
| mmm                          | Month abbreviation. Outputs the month as an abbreviation (e.g., Jan-Dec).  |
| mmmm                         | Month name. Outputs the month as a full name (e.g., January-December).   |
| d                            | Day number. Outputs the day as digits with no leading zero (e.g., 1-9).  |
| dd                           | Day number. Outputs the day as digits with leading zeros (e.g., 01-31).  |
| ddd                          | Day abbreviation. Outputs the day as an abbreviation (e.g., Sun-Sat).  |
| dddd                         | Day name. Outputs the day as a full name (e.g., Sunday-Saturday).  |
| yy                           | Year number. Outputs the year as a two-digit number (e.g., 00-99).   |
| yyyy                         | Year number. Outputs the year as a four-digit number (e.g., 1900-2078).  |
| h                            | Hour number. Outputs the hour as a number without leading zeros (1-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock. |
| hh                           | Hour number. Outputs the hour as a number with leading zeros (01-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.   |
| m                            | Minute number. Outputs the minute as a number without leading zeros (0-59). The m format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.                   |
| mm                           | Minute number. Outputs the minute as a number with leading zeros (00-59). The mm format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.                    |
| s                            | Second number. Outputs the second as a number without leading zeros (0-59).  |
| ss                           | Second number. Outputs the second as a number with leading zeros (00-59).  |
| AM/PM<br>am/pm<br>A/P<br>a/p | 12-hour time. Outputs time using a 12-hour clock. Outputs AM, am, A, or a for times between midnight and noon; outputs PM, pm, P, or p for times from noon until midnight.                                   |

## Trim

**Description:** Removes leading and trailing spaces from a string.

**Syntax:** `Trim ( string )`

| Argument            | Description   |
|---------------------|---|
| <code>string</code> | The <code>string</code> you want returned with leading and trailing |

|  |                |
|--|----------------|
|  | spaces deleted |
|--|----------------|

**Return value:** String. Returns a copy of [string](#) with all leading and trailing spaces deleted if it succeeds and the empty string ("" ) if an error occurs.

---

## Upper

**Description:** Converts all the characters in a string to uppercase.

**Syntax:** [Upper](#) ( [string](#) )

| Argument               | Description   |
|------------------------|---|
| <a href="#">String</a> | The string you want to convert to uppercase letters |

**Return value:** String. Returns [string](#) with lowercase letters changed to uppercase if it succeeds and the empty string ("" ) if an error occurs.

---

## WordCap

**Description:** Sets the first letter of each word in a string to a capital letter and all other letters to lowercase (for example, ROBERT E. LEE would be Robert E. Lee).

**Syntax:** [WordCap](#) ( [string](#) )

| Argument               | Description  |
|------------------------|--|
| <a href="#">string</a> | A string or expression that evaluates to a string with initial capital letters (for example, Monday Morning) |

**Return value:** String. Returns [string](#) with the first letter of each word set to uppercase and the remaining letters lowercase if it succeeds and NULL if an error occurs.

---

## Operators

An operator is a symbol or word in an expression that performs an arithmetic calculation or logical operation; compares numbers, text, or values; or manipulates text strings.

The database buffer supports the following types of operators:

- Arithmetic for numeric data types
- Relational for all data types
- Logical for all data types
- Concatenation for string data types

### Arithmetic operators

These are the arithmetic operators:

| Operator | Meaning        | Example          |
|----------|----------------|------------------|
| +        | Addition       | SubTotal + Tax   |
| -        | Subtraction    | Price - Discount |
| *        | Multiplication | Quantity * Price |
| /        | Division       | Discount / Price |
| ^        | Exponentiation | Rating ^ 2.5     |

### Relational and logical operators

Logical operators can join relational expressions to form more complex boolean expressions.

The result of evaluating a boolean expression is always TRUE or FALSE.

These are the relational and logical operators:

| Operator | Meaning                      | Example                   |
|----------|------------------------------|---------------------------|
| =        | Equals                       | Price = 100               |
| >        | Greater than                 | Price > 100               |
| <        | Less than                    | Price < 100               |
| <>       | Not equal                    | Price <> 100              |
| >=       | Greater than or equal        | Price >= 100              |
| <=       | Less than or equal           | Price <= 100              |
| NOT      | Logical negation             | NOT Price = 100           |
| AND      | Logical and                  | Tax > 3 AND Ship < 5      |
| OR       | Logical or                   | Tax >3 OR Ship < 5        |
| IN       | SQL SELECT IN condition      | Status IN ('A', 'S', 'R') |
| BETWEEN  | SQL SELECT BETWEEN condition | Ship BETWEEN 3 AND 5      |
| LIKE     | SQL SELECT LIKE condition    | Brand LIKE 'Micro%'       |

### Concatenation operator

The concatenation operator joins the contents of two string expressions to form a longer value. You can concatenate strings only.

| Operator | Meaning     | Example       |
|----------|-------------|---------------|
| +        | Concatenate | "cat "+ "dog" |

### Operator precedence in expressions

To ensure predictable results, all operators in an expression are evaluated in a specific order of precedence. When the operators have the same precedence, the database buffer evaluates them left to right.

The following table lists the operators in descending order of precedence:

| Operator            | Purpose  |
|---------------------|--|
| ( )                 | Grouping                                       |
| ^                   | Exponentiation                                 |
| *, /                | Multiplication and division                    |
| +, -                | Addition and subtraction; string concatenation |
| IN, LIKE, BETWEEN   | SQL SELECT statement conditions                |
| NOT                 | Logical negation                               |
| =, >, <, <=, >=, <> | Relational operators                           |
| AND, OR             | Logical AND and logical OR                     |

## Chapter 20: Script Debugger

### Description

Debugging is a process you use to find and resolve errors, or bugs, in your JAL scripts. There are three types of errors you may encounter as your script runs:

- Syntax errors as a result of incorrectly constructed code. You may have forgotten to balance pairs of statements (such as end labels for [LoopWhile](#), [LoopUntil](#), [ForNext](#) loops), or you may have a programming mistake that violates the rules of JAL (such as a misspelled word, missing separator, or argument type mismatch error, mismatched parentheses or an incorrect number of arguments passed to a JAL statement).
- Run-time specific errors after the job starts to run. Examples of run-time errors include attempting an illegal operation, such as dividing by zero or writing to a file that does not exist.
- Logic errors when the script does not perform as intended and produces incorrect results.

To help you isolate all three types of errors and to monitor how your code runs, 24x7 Scheduler provides debugging tools that let you step through your code one line at a time, examine or monitor the values of variables, and trace statement calls.

### Using the Debugger

To use the debugger, set breakpoints in the scripts that you want to examine and click the Start button (or press the F5 key). When the script stops at a breakpoint, you can examine the values of variables and view the call stack. The debugger lets you set breakpoints as well as view and change local script and global variables. You can single-step through the code, continue to the next breakpoint, or skip a few lines and continue running from the cursor location.

### Breakpoints

Breakpoint is a line of code in a job script or user-defined JAL statement at which the debugger automatically suspends execution.

To make the Debugger pause in its execution of your code, you can set a breakpoint:

- In the script text pane, double-click on the desired line of code that is not already a breakpoint.  
Or
- Move the caret to the desired line then either press the F8 key or select the **Toggle Breakpoint** command from the Debug menu.

To remove a breakpoint:

- Double-click the line of code on which the breakpoint has been set.  
Or
- Move the caret to the breakpoint line then either press the F8 key or select the **Toggle Breakpoint** command from the Debug menu.

A breakpoint can be set on a line that contains an executable statement

### Views in the Debugger window

The debugger has two panes. You can resize these at any time.

The top pane shows the full text of a currently executed script. In that pane you can go to a specific line in a script, find a string, print the displayed script, and manage the breakpoints.

The bottom pane contains four tab pages:

|                          |  |
|--------------------------|--|
| <b>Variables Watch -</b> | An expandable list of all the variables in scope. The values in the Variables Watch tab page are updated each time they are changed from a script or Immediate window. You can double-click on any variable to manually change the value of a variable whenever script execution is suspended. |
| <b>Call Stack -</b>      | The sequence of JAL statement calls leading up to the script that was executing at the time of the breakpoint.   |
| <b>Breakpoints -</b>     | A list of breakpoints in the current script.   |
| <b>Immediate -</b>       | A part of the Debugger window in which you can run individual lines of JAL code,   |

|  |   |
|--|---|
|  | usually for the purpose of debugging. The Immediate pane is a kind of scratch pad window in which statements and variables are evaluated immediately. In the Immediate page you can type a statement then press F6 to execute this statement. |
|--|---|

### Step through JAL code

Stepping through your script can help you identify where an error is occurring. You can see whether each line of code produces the results you expected. To step through the code after you have it opened in the Debugger, do one of the following:

- To step through each line of job script and into the code in a user-defined statement called by the main script or by another statement, select the **Debug/Step** command from the Debug menu, or alternatively you can press the CTRL+F5 keys
- To run the code that precedes the desired line of code, and then break so you can step through each line of code, set a breakpoint on that line then select the **Start** or **Continue** command from the Debug menu. The Debugger will suspend execution at the breakpoint line. Use the **Debug/Step** command to continue line by line (see previous paragraph for details).
- To run the rest of the current script, select the **Continue** command from the Debug menu. The Debugger will continue executing script until it reaches a breakpoint or an end of script, whichever is first.
- To skip undesired lines, move the caret to the desired line, select the **Set Next Statement** command from the Debug menu. Select the **Step** or **Continue** command from the Debug window.

The type of stepping you do depends on which portions of code you want to analyze.

## Chapter 21: Testing and Debugging

Sometimes a 24x7 script type job does not behave the way you think it will. Perhaps a variable is not being assigned the value you expect, or a script does not do what you want it to do. In such situations, you can debug the script by using the following options:

- 1 Using the **Debugger**, run your script in debug mode. This allows you to monitor your script execution; evaluate and change variables, and trace the call stack.
- 2 Call the **MessageBox** statement within the script to display the value of a variable.
- 3 Call the **DatabaseDescribe** statement within the script to get definition of the database result set that was created when preparing for data retrieval.
- 4 Enable the **Trace** features then execute the job. The 24x7 Scheduler writes detailed tracing information into the SCRIPT.LOG file, which you can examine using the **Log Viewer**.
- 5 Enable the **Database Trace** features then execute the job. The 24x7 Scheduler writes detailed tracing information for all database calls into the PBTRACE.LOG file. You can examine this using the **Log Viewer**.
- 6 Use the **TelnetConfig** statement to show the Telnet terminal window, which displays all Telnet session messages, sent between the 24x7 Scheduler and the remote host.

After you have found and fixed the problems in your scripts, you should remove all MessageBox statements, so that the job can run without user intervention. You should also disable all tracing features to allow this and other jobs to run at optimum speed without performance hit caused by the tracing.

**See also:**

- Troubleshooting Job Execution
- Troubleshooting Database Connection
- Execution Logs
- Logging Job Execution

## Chapter 22: JAL Examples

The 24x7 Scheduler provides a sample job database that includes examples of 24x7 script type jobs that you can use while you are learning Job Automation Language (JAL). This sample job database is distributed as part of the standard 24x7 installation package.

To see the JAL script for a job:

- 1 Double-click the job icon in the Job Explorer. The Job Wizard dialog box will appear.
- 2 Click the **Next** button, then click the **Edit** button. The JAL Editor window opens with the script in it.
- 3 After reviewing the script, close the Editor, then click the **Next** button again to see the actual job schedule. This is an important part of any scripting job.

Look at the following job examples:

|     |   |
|-----|---|
| 1.  | Running job between 9:00 AM and 5:00 PM every workday.                |
| 2.  | Scheduling job to not to run on particular days (job exception days). |
| 3.  | Converting dates to filenames.  |
| 4.  | Converting comma-separated data file to tab-separated file.           |
| 5.  | Scanning program log file for errors.                                 |
| 6.  | Processing files using file mask.                                     |
| 7.  | Collecting Web server performance statistics.                         |
| 8.  | Monitoring database server free space.                                |
| 9.  | Monitoring file server free space.                                    |
| 10. | Establishing dial-up connection, avoiding line drops.                 |
| 11. | Loading data into database.   |
| 12. | Starting NT service, executing job, stopping service.                 |
| 13. | Paging/notifying system administrators.                               |
| 14. | Restoring network connection.   |
| 15. | Unattended server reboot on demand.                                   |
| 16. | Using FTP commands.   |
| 17. | Redirecting program output to a file.                                 |
| 18. | Verifying overnight data feed.  |
| 19. | Using Windows messages.   |
| 20. | Watching for file changes.  |

### Running job between 9:00 AM and 5:00 PM every workday

```
Dim today, date
Dim time_now, time
Dim is_work_time, boolean
Dim is_holiday, boolean

CHECK_WEEKDAY:
Today( today )
isWeekday( today, is_work_time )
If( is_work_time, CHECK_HOLIDAY, DONE )

CHECK_HOLIDAY:
isHoliday( today, is_holiday )
If( is_holiday, DONE, CHECK_TIME )

CHECK_TIME:
Now( time_now )
isTimeBetween( time_now, 9:00, 17:00, is_work_time )
If( is_work_time, SOMETHING, DONE )

DO_SOMETHING:
// Perform the actual job here
```

```
// ...
```

```
DONE:
```

**See also:**

Other JAL script examples

## Redirecting program output to a file

```
Dim process_id, number
Dim command, string
Dim output, string
Dim position, number
Dim err_found, boolean

// For this example dynamically create batch file that includes just one command
FileSave( "redir.bat" , "ver" )

// Run the batch and terminate it after 30 seconds, in case
// if it does not stop. Batch output is redirected from the
// console to the out.txt file
RunAndWait ( "redir.bat >> out.txt" , "", 30, process_id)

// Read the output file for error checking
FileReadAll ( "out.txt" , output)
Lower (output, output)
Pos (output, "error" , 1, position)
IsGreater (position, 0, err_found)
If (err_found, NOTIFY, END)

NOTIFY:
// Send e-mail message
MailSend("Exchange Settings" , "syspassword" , "admin@mycompany.com" , "Error occurred
while running DOSTEST.BAT" )

END:
// Done
```

**See also:**

Other JAL script examples

## Converting comma-separated data file to tab-separated file

```
Dim pointer, number
Dim buffer, string
Dim found, boolean

// Read comma separated file
FileReadAll( "sometext.txt" , buffer )

// Replace commas with tabs
Pos( buffer, ",", 1, pointer)
isGreater( pointer, 0, found)

LoopWhile( found, ENDLOOP )
    Replace( buffer, pointer, 1, " ", buffer )
    Pos( buffer, ",", pointer, pointer)
```

```
isGreater( pointer, 0, found)
ENDLOOP:

// Write tab separated file
FileSave( "tabtext.txt" , buffer )
```

**See also:**

Other JAL script examples

## Using FTP commands

```
Dim process_id, number
Dim found, boolean

// Watch for file on remote FTP site.
FTPFileExists( "ftp.microsoft.com" , "", "", "disclaimer.txt" , found )

// If the file found,
// continue processing, otherwise exit and wait for the next cycle
if (found, DOWNLOAD, END )

DOWNLOAD:
// Download the file from Microsoft FTP site
FTPGetFile( "ftp.microsoft.com" , "", "", "disclaimer.txt" , "c:\\temp\\disclaimer.txt"
)
// Do something with the downloaded file, for example you can display
// it in the Notepad
Run( "notepad c:\\temp\\disclaimer.txt" , "", process_id )

// Delete the file - in a real-world you most likely will do this
// FTPDeleteFile( "ftp.microsoft.com" , "", "", "disclaimer.txt" )

END:
// Done
```

**See also:**

Other JAL script examples

## Scheduling a job not to run on particular days (exception days)

One of the solutions is to create a text file of exception days so that every time the job runs it compares the current date with days in the exception file. For example, create the "ex\_dates.txt" file using the text that follows:

```
03/01/98
03/08/98
04/01/98
04/08/98
```

Now create the new job with 24x7 script type and schedule this job to run daily. The following script can be used to check dates:

```
Dim buffer, string
Dim found, boolean
Dim today, date
Dim st_today, string
Dim pointer, number
```

```
// Read comma separated file
FileReadAll( "ex_dates.txt" , buffer )

// Get today's date and convert it to a string
// Replace commas with tabs
Today( today )
Format( today, "mm/dd/yy" , st_today)

Pos( buffer, st_today, 1, pointer)
isGreater( pointer, 0, found)

if( found, END, DO_SOMETHING )

DO_SOMETHING:
// Do something here
// ...

END:
```

**See also:**

Other JAL script examples

## Converting dates to file names

Sometimes you may need to calculate file name as a function of date. You can use the following example as a template:

```
Dim today, date
Dim yesterday, date
Dim file_name, string

// Get today's date
Today( today )
// Calculate yesterday's date
DateAdd( today, -1, yesterday )
// Convert to string in mmddyyyy format
Format( yesterday, "mmddyyyy" , file_name )
// Append file extension
Concat( file_name, ".dat" , file_name )

// Do something with this file,
// for example FTP, Copy, Load into database, etc.
// ...
```

In some cases, you may use available macro-parameters to simplify date calculations. For example, the following script produces the same end-result as the script above:

```
Dim file_name, string

// Build file name
Concat( @DP" mmddyyyy" , ".dat" , file_name )

// Do something with this file,
// for example FTP, Copy, Load into database, etc.
// ...
```

**See also:**[Other JAL script examples](#)

## Scanning program log file for errors

```
Dim buffer, string
Dim position, number
Dim err_found, boolean

// Load the log file into memory
FileReadAll ("program.log" , buffer)
// Convert to lower case to find all sorts of errors
Lower (buffer, buffer)
// Search for word "error"
Pos (buffer, "error" , 1, position)
// If error found, notify system administrator
IsGreater (position, 0, err_found)
If (err_found, NOTIFY, END)

NOTIFY:
// Send e-mail message
MailSend( "Exchange Settings" , "syspassword" , "admin@mycompany.com" , "Error
occurred while running PROGRAM.EXE. See program log file for details." )

END:
// Done
```

**See also:**[Other JAL script examples](#)

## Processing files using file mask

```
Dim file_name, string
Dim found, boolean
Dim attr, number
Dim read_only, boolean
Dim hidden, boolean
Dim system, boolean
Dim directory, boolean
Dim archived, boolean
Dim message, string

// Start file search
FileFindFirst( "*.txt" , file_name, found )

LoopWhile( found, ENDLOOP )
    // File found, do something with the file here
    // ...
    // For example we can get file attributes then display them
    FileGetAttr( file_name, attr )
    BitwiseGetBit( attr, 1, read_only )
    BitwiseGetBit( attr, 2, hidden )
    BitwiseGetBit( attr, 3, system )
    BitwiseGetBit( attr, 5, directory )
    BitwiseGetBit( attr, 6, archived )

    // skip directories
    IfThen( directory, FIND_NEXT )
```

```
// Show nice message
Char( 13, new_line )
Concat( "File: ", file_name, message )
Concat( message, "\n\tRead Only: ", message )
Concat( message, read_only, message )
Concat( message, "\n\tHidden: ", message )
Concat( message, hidden, message )
Concat( message, "\n\tSystem: ", message )
Concat( message, system, message )
Concat( message, "\n\tDirectory: ", message )
Concat( message, directory, message )
Concat( message, "\n\tArchived: ", message )
Concat( message, archived, message )
MessageBox( message )

FIND_NEXT:
// Find next file
FileFindNext( file_name, found )
ENDLOOP:
```

**See also:**

Other JAL script examples

## Collecting Web server performance statistics

```
// This script allows checking Web server response time.
// The number is saved in a file, which can be helpful
// in analyzing Web server performance such as estimating
// average response time and building busy rate graph as
// a dependency of time of day.

Dim current_time, time
Dim current_date, date
Dim start_time, datetime
Dim end_time, datetime
Dim duration, number
Dim file_number, number
Dim line, string

// Get start time
Today( current_date )
Now( current_time )
DateTime( current_date, current_time, start_time )

// Download index page from a Web server
// For example use popular Microsoft Web site
WebGetPageHTML( "http://home.microsoft.com/", "c:\\temp\\msweb.tmp" )

// Get end time
Today( current_date )
Now( current_time )
DateTime( current_date, current_time, end_time )

// Calculate response time
DateTimeDiff( start_time, end_time, duration )

// Build text line
Concat( "Start: ", start_time, line )
Concat( line, " End: ", line )
Concat( line, end_time, line )
Concat( line, " Duration: ", line )
Concat( line, duration, line )
```

```
// Append built line to the statistics file
FileOpen( "webstat.txt" , "LineMode" , "Write" , TRUE, file_number )
FileWrite( file_number, line )
FileClose( file_number )

// Uncomment next line to open statistics file in Notepad
// Run( "notepad webstat.txt" , "", file_number )
```

**See also:**

Other JAL script examples

## Monitoring file-server free space

```
// Before trying this script, correct the
// network drive letter specified below, if necessary

Dim drive, string
Dim free_space, number
Dim low_space, boolean

Set( drive, "K" )
// Get free space
// Windows95 OSR-1 users use DiskGetFreeSpace instead of DiskGetFreeSpaceEx
DiskGetFreeSpaceEx( drive, free_space )
// Check threshold - 50 Mbytes = 50 * 2^20
isLess( free_space, 52428800, low_space )

if( low_space, PROBLEM, END )
PROBLEM:
// Page network administrator
MailSend( "Exchange Settings" , "system" , "12345678@pagenet.com" , "Low of space" ,
"Free space on drive K below 50 Mbytes" )

END:
```

**See also:**

Other JAL script examples

## Using Windows messages

```
// This example was developed to demonstrate how to send
// Windows messages from a script.
Dim process_id, number
Dim window_handle, number
Dim edit_handle, number

// Use Notepad for demonstration
// Run Notepad
Run( "notepad" , "", process_id )
// Wait 2 seconds for the Notepad to open
Wait( 2 )
// Get handle of the Notepad window
ProcessGetWindow( process_id, window_handle )
// Get handle of the edit box
WindowGetChild( window_handle, edit_handle )
// Send EM_SETREADONLY message to the edit box
// to make it read only
WindowSendMessage( edit_handle, 207, 1, 0 )
```

```
// Open schedule log file in the Notepad
WindowActivate( window_handle )
SendKeys( "{ALT}FOschedule.log{ENTER}" )
```

**See also:**

Other JAL script examples

## Unattended server reboot on demand

```
// In this example, we assume that the 24x7 Scheduler is running on the server
// and that this job is setup as an "e-mail watch." On receiving the specified // e-
// mail message, 24x7 Scheduler will reboot the computer. This script has
// just one line:
```

Reboot

```
// That's all, folks! Now imagine how easy it is to reboot your database and
// Web servers without leaving your home. You can also setup this job to run
// every day so that it will restart the server every morning at 7:00 AM before
// people come to the office.
```

**See also:**

Other JAL script examples

## Restoring network connection

```
// Use Windows Explorer to restore a disconnected network drive before
// running a program that accesses data files on the disconnected drive.
```

```
Dim process_id, number
Dim window_handle, number
Dim running, boolean
```

```
Run( "explorer k:\data" , "", process_id )
// Wait 10 seconds to allow the Explorer to restore connection
Wait( 10 )
```

```
// First let's try to close Explorer gracefully (this is always better than killing
// the process.)
// However, using the ProcessKill statement guarantees that the process termination. A
// keystroke, on the other hand,
// is sent to an active window, which can change at anytime.
SendKeys( "{ALT}{F4}" )
```

```
// Wait 2 more seconds
Wait( 2 )
// Check whether Explorer is still running or not
ProcessGetWindow( process_id, window_handle )
isGreater( window_handle, 0, running)
```

```
if( running, KILL, DO_MAIN_JOB )
KILL:
ProcessKill( process_id )
```

```
DO_MAIN_JOB:
// Do the main job here
// Run ...
```

**See also:**[Other JAL script examples](#)

## Watching for file changes

```
// This is an example of a "file change watch" job. A regular "file watch" job checks
// file presence only. In some situations, you may need to run a job only when
// a change occurs.
```

```
// This script uses the "change.txt" file to store "watch file" attributes between
runs.
```

```
Dim buffer, string
Dim file_time, string
Dim file_date, string
Dim date_time, datetime
Dim buffer2, string
Dim no_change, boolean

// Read parameter file into buffer
FileReadAll( "change.txt" , buffer )

// Get watch file date/time
FileTime( "k:\data\account.dat" , file_time )
FileDate( "k:\data\account.dat" , file_date )
DateTime( file_date, file_time, date_time )
// Convert to string
Format( date_time, "mm/dd/yyyy hh:mm:ss" , buffer2 )
// Compare file times
isEqual( buffer, buffer2, no_change )

// If the file date/time is the same, then nothing to do, otherwise
// update the parameter file and perform the main job.
If( no_change, END, UPDATE )
END:
Exit

UPDATE:
FileSave( "change.txt" , buffer2 )
// do main job here
// ...
```

**See also:**[Other JAL script examples](#)

## Paging/notifying system administrators

The 24x7 Scheduler allows you to setup a job that can send a page or an e-mail message to one recipient at a time. Sometimes, if a job fails, it is necessary to notify more than one person. You can use JAL to add this function to virtually any job. One way to do this is to setup the "file type" notification action for the main "program" or "database" type job.. In the event of a failure, a semaphore file will be created (for example: job.err). Another job of 24x7 script type instantly checks presence of the specified semaphore file. As soon as the file is detected, the job deletes the file and sends pages or e-mail messages to the specified recipients. Note that the second job will need to be scheduled as a "file watch". The following example shows how to send multiple messages by using JAL e-mail statements.

```
// Note: The list of recipients can be hard-coded to simplify scripting.
// This list of recipients can be read from file. This was done in order to make the
// script reusable.
// Thus recipients can be changed without changing the job definition.

Dim file_number, number
Dim recipient, string
Dim end_of_file, boolean

// Open recipients file for reading line by line
FileOpen( "mail.lst" , "LineMode" , "Read" , "", file_number )

// Read the file line by line until the end of file.
// Each recipient must be on a separate line.
EOF( file_number, end_of_file )
LoopUntil end_of_file, ENDLOOP
    FileRead( file_number, recipient )
    // Send e-mail
    MailSend( "Exchange Settings" , "syspassword" , recipient, "any subject" , "any
message" )

    EOF( file_number, end_of_file )
ENDLOOP:

// Done
FileClose( file_number )
```

**Notes:**

- There are many companies that provide reliable “e-mail to pager” services. For example: CompuServe, SkyTel, PageNet, MobileComm, and PageMart. There is no major difference between sending e-mail messages and sending messages to a pager. The only real difference is that when you send a message to a pager you must use a special addressing format. For example, the address format for the CompuServe is: [CompuServe\\_ID@mobile.compuserve.com](mailto:CompuServe_ID@mobile.compuserve.com). If the account number (CompuServe ID) is 79999.9999, the recipient could receive wireless e-mail by using the address [79999.9999@mobile.compuserve.com](mailto:79999.9999@mobile.compuserve.com). Note that a period must be used in place of a comma.
- In the case of an alphanumeric message, the number of characters received by the recipients depends on how they setup their pager preferences. Keeping your message short and concise helps to ensure that the recipient gets the most out of your message.

**See also:**

Other JAL script examples

## Monitoring database free space

This example shows how easily you can perform various database operations in 24x7 Scheduler. To simplify the script we saved the SQL part in a separate file called [free\\_space.sql](#). The script will load this file and dynamically execute the loaded SQL. This SQL was designed for Oracle 7 databases. You may need to customize it for your database.

```
// Load and execute SQL script from "free_space.sql" file
// If the result is positive (at least one segment found),
// notify the database administrator about potential problems.

Dim SQL, string
Dim rows, number
Dim problem, boolean
Dim message, string
```

```
// Load file
FileReadAll( "free_space.sql" , SQL )
// Connect to database and retrieve "bad" segments,
// see "free_space.sql" file for details
DatabaseConnect( "Sales DB" )
DatabaseRetrieve( SQL, rows )
DatabaseDisconnect

isGreater( rows, 0, problem )

// If problem detected, notify DBA
If( problem, NOTIFY, END )

NOTIFY:
// Save retrieved data in the temporary file
DatabaseSave( "c:\\temp\\message.tmp" , "TXT" , rows )
// Read temp. file contents and then e-mail to DBA
FileReadAll( "c:\\temp\\message.tmp" , message )
MailSend( "Exchange Settings" , "pwrdr" , "ora_dba@my_company.com" , "Database Problem"
, message )
// Delete temp. file
FileDelete( "c:\\temp\\message.tmp" )

END:
```

## FREE\_SPACE.SQL

```
SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
      'Name: ' || seg.segment_name || chr(13) || chr(10) ||
      'Type: ' || seg.segment_type || chr(13) || chr(10) ||
      'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
      'Next extent size: ' || to_char(seg.max_extents, '999,999,999') || chr(13) ||
chr(10) ||
      'Problem: Max extent reached'
FROM sys.dba_segments seg
WHERE seg.extents = seg.max_extents

UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
      'Name: ' || seg.segment_name || chr(13) || chr(10) ||
      'Type: ' || seg.segment_type || chr(13) || chr(10) ||
      'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
      'Next extent size: ' || to_char(t.next_extent, '999,999,999') || chr(13) || chr(10)
||
      'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
     sys.dba_tables t
WHERE seg.segment_type = 'TABLE'
      AND seg.segment_name = t.table_name
      AND seg.owner = t.owner
      AND NOT EXISTS (SELECT tablespace_name
                      FROM dba_free_space free
                      WHERE free.tablespace_name = t.tablespace_name
                        AND free.bytes >= t.next_extent
                      )

UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
      'Name: ' || seg.segment_name || chr(13) || chr(10) ||
      'Type: ' || seg.segment_type || chr(13) || chr(10) ||
      'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
```

```
'Next extent size: ' || to_char(i.next_extent, '999,999,999') || chr(13) || chr(10)
||
'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
     sys.dba_indexes i
WHERE seg.segment_type = 'INDEX'
      AND seg.segment_name = i.index_name
      AND seg.owner = i.owner
      AND NOT EXISTS (SELECT tablespace_name
                      FROM dba_free_space free
                      WHERE free.tablespace_name = i.tablespace_name
                      AND free.bytes >= i.next_extent
                      )
UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
      'Name: ' || seg.segment_name || chr(13) || chr(10) ||
      'Type: ' || seg.segment_type || chr(13) || chr(10) ||
      'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
      'Next extent size: ' || to_char(c.next_extent, '999,999,999') || chr(13) || chr(10)
||
'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
     sys.dba_clusters c
WHERE seg.segment_type = 'CLUSTER'
      AND seg.segment_name = c.cluster_name
      AND seg.owner = c.owner
      AND NOT EXISTS (SELECT tablespace_name
                      FROM dba_free_space free
                      WHERE free.tablespace_name = c.tablespace_name
                      AND free.bytes >= c.next_extent
                      )
UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
      'Name: ' || seg.segment_name || chr(13) || chr(10) ||
      'Type: ' || seg.segment_type || chr(13) || chr(10) ||
      'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
      'Next extent size: ' || to_char(r.next_extent, '999,999,999') || chr(13) || chr(10)
||
'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
     sys.dba_rollback_segs r
WHERE seg.segment_type = 'ROLLBACK'
      AND seg.segment_name = r.segment_name
      AND seg.owner = r.owner
      AND NOT EXISTS (SELECT tablespace_name
                      FROM dba_free_space free
                      WHERE free.tablespace_name = r.tablespace_name
                      AND free.bytes >= r.next_extent
                      )
)
```

**See also:**

Other JAL script examples

## Starting NT service, executing job, stopping NT service

Some Windows NT applications are designed as Windows NT services. In some cases it is not necessary to keep such application running all the time, as that application will take some computer and Operation System resources leaving less resources for other applications. This example shows how to manage the Windows NT service from JAL.

```
// This script starts Personal Oracle 8.0, runs a program
// that loads information into the database, then the script
// shutdowns the database.

// Start Oracle database. Note that service OracleStartORCL automatically
// starts another service OracleService.
ServiceStart "OracleStartORCL"
ServiceStart "OracleTNSListener80"
ServiceStart "OracleClientCache80"

// Do main job here
// ... Run ...

// Shutdown database
ServiceStop "OracleClientCache80"
ServiceStop "OracleTNSListener80"
ServiceStop "OracleStartORCL"
ServiceStop "OracleService"
```

**See also:**

Other JAL script examples

## Loading data into database

```
// This script imports data into a temporary table called TEMP_SALES_ORDER,
// then populates the main SALES_ORDER table from the temporary table.
Dim imported_rows, number
Dim old_rows, number
Dim new_rows, number
Dim message, string

// Connect to database
DatabaseConnect( "Sales DB (Oracle 7.2)" )
// Prepare temp.table
DatabaseExecute( "TRUNCATE TABLE scott.temp_sales_order" , old_rows )
// Populate temp. table
DatabaseImport( "scott.temp_sales_order" , "order.txt" , imported_rows )
// Delete from main table matching order to avoid duplicate key problem
DatabaseExecute( "DELETE FROM scott.sales_order WHERE id IN (SELECT id FROM
scott.temp_sales_order)", old_rows )
// Populate main table
DatabaseExecute( "INSERT INTO scott.sales_order SELECT * FROM scott.temp_sales_order"
, imported_rows )
// Disconnect from database and commit transaction
DatabaseDisconnect

// Notify job owner about results
Subtract( imported_rows, old_rows, new_rows )
Concat( "Updated orders: ", old_rows, message)
Concat( message, ", New orders: ", message)
Concat( message, new_rows, message)

MailSend( "Exchange Settings" , "pswrld" , "sctott@mycompany.com" , "Sales Order
Update" , message )
```

**See also:**

Other JAL script examples

## Establishing dial-up connection, avoiding line drops

```
// This script demonstrates how to initiate a dial-up connection from JAL
// script. To use a different connection name, replace CompuServe with
// the name of your dial-up connection. RnaDial and your connection name
// are case-sensitive, so match the text exactly. Make sure to setup your
// dial-up connection so that it remembers the password and does not have to prompt
// for it.

Dim process_id, number
Dim window_handle, number

// Connect to CompuServe
Run( "rundll rnaui.dll,RnaDial CompuServe" , "", process_id )
// Wait 60 seconds to allow dial-up networking to initiate the connection
Wait( 60 )
// do something here
// ...
// done

// To disconnect from CompuServe, close the dial-up window.
// To leave the connection online, comment the following two lines
WindowFind( "Connect To%", window_handle )
WindowClose( window_handle )

// If you do not want to close the connection, but keep this
// connection online, even with long periods of inactivity, you can use
// the ping command. setup a new job that will run All Day every minute.
// Select the "program" type for this job and specify the ping command in the
// program name field. This will help avoid line drops.
// For example: ping www.ibm.com
```

### See also:

Other JAL script examples

## Verifying overnight data feed

```
// Check data files that arrived earlier.
Dim file_date, date
Dim file_time, time
Dim today, date
Dim now, time
Dim not_found, boolean
Dim file_OK, boolean
Dim date_OK, boolean
Dim time_OK, boolean
Dim file_name, string

Today( today )
Now( now )

CHECK_ACCOUNTS:
// Get date/time for Accounts
Set( file_name, "j:\data\account.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )
```

```
// Check if it is file_OK
IsEqual( file_date, today, date_OK )
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, CHECK_HOLDINGS, BAD_FILE )

CHECK_HOLDINGS:
// Get date/time for Holdings
Set( file_name, "j:\data\holding.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )

// Check if it is file_OK
IsEqual( file_date, today, date_OK )
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, CHECK_SECURITIES, BAD_FILE )

CHECK_SECURITIES:
// Get date/time for securities
Set( file_name, "j:\data\secirity.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )

// Check if it is file_OK
IsEqual( file_date, today, date_OK )
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, DONE, BAD_FILE )

BAD_FILE:
// Inform operation personal about bad file
MailSend( "Exchange Settings" , "pswrd" , "david@mycompany.com" , file_name,
"Overnight feed failed. File specified in the subject was not updated!" )
MailSend( "Exchange Settings" , "pswrd" , "joe@mycompany.com" , file_name, "Overnight
feed failed. File specified in the subject was not updated!" )

DONE:
```

## Chapter 23: Technical Support

Your questions, comments, and suggestions are welcome.

For technical support, e-mail to [support@softtreetech.com](mailto:support@softtreetech.com) or use the on-line support form at <http://www.SoftTreeTech.com/Support.htm>.

When reporting problems, please provide as much information as possible about your problem. Be sure to include the following information:

- 1 Is the problem reproducible? If so, how?
- 2 What version of Windows are you running? For example, Windows 95, Windows NT 4.0, etc.
- 3 What version of the 24x7 Scheduler are you running?
- 4 If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.
- 5 If the problem involves an external program, provide as much information as possible about this program.
- 6 Make sure you include the serial number for your copy of 24x7 Scheduler . Use the **Help/About** menu to look up the correct numbers. Registered users have priority support.

For registration information, purchasing or other sales information, please contact our sales department: [sales@softtreetech.com](mailto:sales@softtreetech.com).

For general information, software updates, the latest information on known problems and answers to frequently asked questions, visit the 24x7 Scheduler home page on the Web: <http://www.SoftTreeTech.com/24x7/>.

We are happy to help in any way we can, but if you are having problems, please check the troubleshooting section first to see if your question is answered there.

## Chapter 24: Frequently Asked Questions (FAQ)

**Q: I have setup a job that should run unattended. This job should send an email messages in case of an error. When the job fails, it creates an email message, but the message is not sent and the "New message" window remains on the screen. How do I fix it?**

**A:** You are using MAPI email interface. Different email programs utilize different MAPI message properties. That's why 24x7 Scheduler provides configuration options that allow you to tune it for your email program. The email will not be send if you have incompatible properties selected. To fix this problem, select Tools/Options menu. The Options dialog will appear. Change email send method to "send by name". Either "send by name" or "send by address" option should be compatible with your email program.

**Q: After I have created a new job and then clicked the Save button on the toolbar, the job became disabled. What's wrong?**

**A:** You did not specify at least one mandatory property for that job. See job log for details on invalid properties. Most likely, you forgot to specify the job start time. Double-click the job's icon, then correct the job schedule. Release the Disable button on the toolbar. Save changes.

**Q: How can I setup my program to run every workday from 9:00 AM to 5:00 PM?**

**A:** Create a new job of the 24x7 script type. Schedule this job to run "All Day". Put in the script some logic to check the desired time condition. If the condition is satisfied, use the Run statement to start the desired program.

**Q: How can I setup a "Fix It" job?**

**A:** Activate the "Error" Notify Event for the primary job. setup either "File" or "E-mail" Notify Action. setup your "Fix It" job as a "file watch" or "E-mail watch" correspondingly. Specify "watch" condition matching parameters of the Notify Action that the 24x7 Scheduler will perform in case the primary job fails.

**Q: I am using just one computer and I want to run some of my "long running" database and 24x7 script jobs in the background. How do I do that?**

**A:** Turn on the "asynchronous job" property.

**Q: Is there a way to save the result returned by a database job?**

**A:** Do not use a database type job for this sort of task. Instead, create a 24x7 script type job. Use DatabaseConnect, DatabaseRetrieve, DatabaseSave, and DatabaseDisconnect statements to connect to the database, retrieve data, save the data in an external file, and then end the database connection.

**Q: After each job execution, 24x7 Scheduler leaks some memory. Why is that?**

**A:** For performance reasons 24x7 Scheduler keeps a copy of the job execution log loaded into computer memory. While executing a job, the 24x7 Scheduler may create new entries in the job execution log. These new entries take some additional memory chunk, until the log reaches the maximum number of allowed entries specified in the system preferences. After that, the most-recent entry pushes the oldest entry out from the log. The memory "leakage" process will then stop. In order to minimize the amount of memory required for the job log you may want to do the following:

- a. Decrease the maximum number of allowed log entries.
- b. Disable logging for jobs that run often and it is not important to you whether such jobs were successful or not.

**Q: I have difficulties running old DOS applications asynchronously. As soon as the screen saver takes over, my DOS job stops until I click on the DOS window!**

**A:** Screen Savers prevent DOS Applications from running in the background. One way to avoid this problem is to use a PIF file to run the application. In the PIF file editor check the "Allow background processing" and "run in a window". This allows the application to act more like a window and continue its process.

**Q: I have installed 24x7 Scheduler on my new server. Can I copy all existing settings from the old installation?**

**A:** Yes! See **Installation and Uninstallation** topic for details.

**Q: How does 24x7 Scheduler know if something ran successfully?**

**A:** When a program job is setup to run synchronously, 24x7 Scheduler checks the "Exit Code" for that program after the job completes. Exit Code values are used by many programs to indicate if they ran successfully. There is no way for the 24x7 Scheduler to find out an Exit Code for the program that you setup to run asynchronously. The 24x7 Scheduler simply manages to start such program and does not know when and how the program completes. Some programs create an application log file during the programs run. You may also try to run such programs using 24x7 script, then search the application log file for common error messages. For a database job, 24x7 Scheduler always checks the "SQL Code" that the database returns after processing. This code reports whether the performed database operation ran successfully or not.

**Q: I would like to run several programs sequentially in one job. How do I set it up?**

**A:** There are two ways to do this. One way is to create a .BAT file that runs all the desired application one by one, then schedule this .BAT file as a single job. Another way is to create a 24x7 script type job, then use the JobRunAndWait statement to implement the desired batch logic.

**Q: I have scheduled a job that kicks off the DOS BAT file. After the .BAT file completes, the DOS window remains open. How can I avoid this?**

**A:** You have two options:

- Run this BAT file manually. Select **Properties** from the window's control menu. Check the **Close on Exit** option.
- Specify the command line for this program as `CMD.EXE /c C:\MYPATH\MYFILE.BAT`. The /c tells the system CMD.EXE command to exit after processing the MYFILE.BAT file. For more help on the CMD command, type `CMD /?` at the command prompt. On Windows 95/98 use the command `COMMAND` instead of `CMD`.

**Q: How do I know which job and when the 24x7 Scheduler will run next?**

**A:** You can rest the mouse pointer over the third section of the Status Bar to view this information. Double-click on this section to locate and highlight the very first pending job. For more details see **Job Explorer** topic. You can also use the **Job Monitor** to get the list of all pending jobs in the 24-hour timeframe.

**Q: How can I configure a program to run after the data it depends on is available?**

**A:** Setup this job schedule as a "file watch". Specify one or more data files in the **semaphore files** property. Set a reasonable polling interval at which the 24x7 Scheduler will check for the presence of specified files.

**Q: I have two 24x7 Schedulers installed in two offices. One of them is watching for a file. As soon as the file arrives, I would like it to be e-mailed to another office where the second scheduler will upload this file to a database. How can I do that?**

- A:** Setup a "file watch" job using "Run 24x7 Script". job type Code the SendMailWithAttachment statement to transmit the file to another office. In the destination office, setup an "e-mail watch" job of "Run 24x7 Script" type. Code the "upload" logic using available file and database statements.

# Index

/

/AGENT, 61  
/JOB, 45  
/NOTIMER, 48  
/RCONTROL, 83  
/SCRIPT, 84

@

@ prefix, 98  
@SCRIPT, 99

2

24x7 /AGENT, 61  
24x7 Remote Control, 81

A

About Fail-over Mode, 55  
About Remote Agents, 60  
Access, 61  
Account, 36  
ACCOUNT, 87  
Actions, 18  
add a new profile, 62  
Add profile, 82  
Add/Remove, 26  
Add/Remove Programs, 26  
Adding New Job, 40  
Adding new user- defined statement, 100  
AGENT, 87  
AgentTest, 196  
Apply, 51  
ASCII, 151  
ASCII character, 99  
ASYNCR, 87  
asynchronous, 60  
Asynchronous Operations, 28  
Asynchronous Process, 32  
Attach to remote host, 82  
Attach to Remote Host, 82  
Attachment, 36  
Attachments, 21

B

Backup Host, 32  
BACKUP\_AGENT, 87  
BACKUP\_HOST, 88  
Bitwise statements, 110  
BitwiseAnd, 110  
BitwiseClearBit, 111  
BitwiseFlipBit, 111  
BitwiseGetBit, 112  
BitwiseNot, 113  
BitwiseOr, 112  
BitwiseSetBit, 113

BitwiseXor, 114  
Browse, 13, 70  
BufSize=n, 57

C

Call, 197  
Case, 107, 266  
CaseElse, 107  
Caution  
Ceiling, 205, 271  
Changes to the System Time, 20  
Char, 231, 278  
ChooseCase, 107  
Clipboard statements, 114  
ClipboardGet, 114  
ClipboardSet, 115  
COMMAND, 88  
command line parameters, 32, 216, 217, 219  
Concat, 232  
ConcatEx, 232  
Connect, 62, 63, 82  
Control Panel, 26, 67  
Control-of-Flow Statements, 102  
Converting dates to file names, 297  
Converting files, 150  
Copy, 75  
Copying, Pasting and Cutting Text, 75  
Create semaphore file(s), 38  
Creating database profiles, 67  
Creating Job Shortcut, 45

D

Daily and Weekly Schedule, 35  
Data type checking fnctions, 257  
Database, 79  
Database Commands, 17  
Database Interfaces, 27  
Database Jobs, 66  
Database Manager, 43  
Database Profiles, 28, 67  
Database statements, 115  
DatabaseConnect, 115  
DatabaseConnectEx, 116  
DatabaseCopy, 117  
DatabaseDelete, 117  
DatabaseDescribe, 118  
DatabaseDisconnect, 118  
DatabaseExecute, 118  
DatabaseExport, 119  
DatabaseGet, 119  
DatabaseImport, 120  
DatabaseInsert, 120  
DatabasePaste, 121  
DatabasePipe, 121  
DatabaseRetrieve, 123  
DatabaseRowCount, 123  
DatabaseSave, 124  
DatabaseSet, 124  
DatabaseSetFilter, 125

DatabaseSetSort, 126  
DatabaseSetSQLSelect, 127  
DatabaseUpdate, 127  
Date and time functions, 259  
Date and time statements, 128  
DateAdd, 130  
DateDiff, 130  
DateTime, 129, 260  
DateTimeAdd, 131  
DateTimeDiff, 131  
DateTimePart, 132  
DAY\_NAME, 88  
DAY\_NUMBER, 88  
DayName, 132, 261  
DayNumber, 133, 261  
DaysAfter, 261  
DDE statements, 137  
DDEClose, 137  
DDEExecute, 138  
DDEGetData, 138  
DDEOpen, 139  
DDESetData, 139  
Defining the ODBC data source, 66  
DEL, 84  
Delay, 33  
DELAY, 88  
Delete, 63, 75, 84  
DELETE, 84  
delete a profile, 63  
Delete profile, 82  
Delete Semaphore File Rule, 36  
DELETE\_RULE, 88  
Deleting Job, 40  
Deleting the text, 75  
Deleting user- defined statement, 101  
Dependencies, 50  
Dependencies Diagram, 50  
DESCRIPTION, 88  
Dir, 140, 168  
DirWaitForUpdate, 141  
disable/enable a job, 41  
Disabled, 40  
DISABLED, 88  
Disabling/Enabling Job, 40  
DiskGetFreeSpace, 198  
DiskGetFreeSpaceEx, 199  
Divide, 206  
Drag and Drop Interface, 18  
Driver, 56

## E

EBCDIC, 151  
Edit, 63  
Edit/Find, 73  
Edit/Go to Bookmark, 74  
Edit/Go to Line, 74  
Edit/Replace, 74  
Editor, 73, 100  
E-mail interface, 21  
E-mail interface library, 22  
Enabled, 47  
End Date, 35  
End Process, 45  
End Time, 35

END\_DATE, 88  
END\_TIME, 88  
Entering JAL and SQL scripts, 73  
EOF, 141  
Error Messages, 44  
EXAMPLE.JDL, 91  
Examples, 126  
Exception Dates, 72  
Execute SQL command(s), 37  
Execution Logs, 19  
Exit, 14, 103  
Exit Code, 220  
Explorer, 15  
Extension manager, 22  
External Interfaces Overview, 84

## F

Fact, 272  
Fail-over, 55, 79  
Fail-over Mode, 26, 55  
FILE, 88  
File conversion, 150  
File replication and synhronization statements, 164  
File statements, 140  
File Type Details, 18  
File Types, 17  
File/Export, 74  
File/Import, 74  
FileClose, 142  
FileConvert, 150  
FileCopy, 143  
FileCopyEx, 143  
FileDate, 145  
FileDelete, 146  
FileDeleteEx, 147  
FileExists, 142  
FileFindFirst, 147  
FileFindNext, 148  
FileGetAttr, 152  
FileGetPos, 153  
FileMove, 144  
FileMoveEx, 145  
FileOpen, 154  
FilePrint, 155, 210  
FileRead, 156  
FileReadAll, 156  
FileReadLine, 157  
FileRename, 157  
FileReplaceEx, 149  
FileSave, 158  
FileSearchEx, 148  
FileSetAttr, 152  
FileSetPos, 154  
FileSize, 159  
FileSplitName, 158  
FileTime, 146  
FileTransfer, 144  
FileUnzip, 161  
FileWrite, 159  
FileZip, 160  
Fill, 233, 278  
Find, 73  
Find Next, 74  
Find What, 74

Floor, 206  
FOLDER, 88  
Format, 39, 233  
Format Masks, 39  
Format Symbol, 286  
ForNext, 104  
FREE\_SPACE.SQL, 304  
Friday, 88  
FRIDAY, 88  
FTP statements, 170  
FTPConfig, 170, 173, 174  
FTPDeleteFile, 171  
FTPDDir, 168, 171  
FTPFileExists, 172  
FTPGetFile, 173  
FTPPutFile, 174  
FTPRenameFile, 174

## G

Generate status report, 70  
GetRow, 267  
Getting Started, 11  
GetToken, 234  
Go to Command, 74  
GoTo, 105  
Graphical Dependencies Editor, 50

## H

hidden, 12, 32  
Holidays, 22, 72  
Host, 32  
HOST, 88  
Hot Keys, 77  
Hour, 262

## I

ID, 88  
Identifier, 38  
IfThen, 106  
Ignore Errors, 34  
IGNORE\_ERRORS, 88  
Importing and Exporting Scripts, 74  
IniFileGetKey, 161  
IniFileSetKey, 162  
Init. Timeout, 33  
INIT\_TIMEOUT, 88  
input file, 19  
Input files, 19  
Installation, 24  
Installing Remote Agents, 25  
InStr, 234  
Integer, 272  
Interface, 79  
Internet statements, 181  
INTERVAL, 88  
Inverse  
IsDate, 187, 257  
IsDateBetween, 187  
IsEqual, 188  
IsGreater, 188  
IsGreaterOrEqual, 189

IsHoliday, 189  
IsLess, 190  
IsLessOrEqual, 190  
IsNull, 258  
IsNumber, 191, 258  
IsRowModified, 267  
IsRowNew, 268  
IsTaskRunning, 212  
IsTime, 191, 259  
IsTimeBetween, 192  
IsWeekday, 192  
IsWeekend, 192

## J

JAL Examples, 294  
JDL Properties, 87  
JDL.BAT, 91  
Job Automation Language, 97  
Job Automation Scripts, 17  
Job Dependencies, 50  
Job Explorer, 15, 43  
Job Log, 79  
Job management statements, 175  
Job Monitor, 80  
Job Properties, 31  
Job Type, 31  
Job Types, 16  
Job Wizard, 30, 41  
JOB\_TYPE, 88  
JobCreate, 175  
JobDelete, 176  
JobDescribe, 176  
JobEnable, 177  
JobGetStatus, 178  
JobModify, 179  
JobRemoteRun, 180  
JobRun, 179  
Jobs, 15  
Jobs Database, 21

## K

Keystroke, 33  
KEYSTROKE, 88

## L

Left, 235, 279  
LeftTrim, 279  
Length, 235  
Loading data into database, 306  
Local, 57  
Location, 57  
LOG, 88  
Log Viewer, 79, 82  
LogAddMessage, 199  
LogAddMessageEx, 200  
Logging Enabled, 33  
Logical statements, 186  
LogSearch, 201  
LogSearchEx, 201  
LogTen, 273  
LogWaitForUpdate, 202

Long, 273  
LoopUntil, 109  
LoopWhile, 108  
Lotus Notes, 21  
Lotus Notes extension manager, 22  
Lower, 236, 280  
LTrim, 236

## M

Macro-parameters, 32, 38  
macro-variables, 98  
mail statements, 21  
Mail statements, 194  
MailSend, 194  
MailSendWithAttachment, 195  
MakeDate, 133  
MakeDateTime, 134  
MakeTime, 134  
Managing Remote Jobs and Configurations, 82  
MAPI, 21  
Master, 58, 79  
Master Scheduler, 81  
Match, 237, 280  
Match Case, 74  
Match Whole Word Only, 74  
Matches, 282  
MaxListeningThreads=n, 57  
MaxRetry=n, 57  
MemoryGetFree, 203  
MESSAGE, 88  
Message Text, 36  
MessageBox, 204  
Metacharacter, 281  
metacharacters, 237, 281  
Minimum Hardware Requirements, 26  
Minute, 262  
Miscellaneous functions, 265  
Miscellaneous statements, 196  
modify an existing profile, 63  
Modify profile, 82  
Modifying user- defined statement, 101  
Monday, 88  
MONDAY, 88  
Monitoring database free space, 303  
Monitoring jobs, 80  
Month, 262  
Monthly Schedule, 35  
MONTHLY\_TYPE, 88  
Moving Job to Another Folder, 41  
MSG\_ACCOUNT, 88  
MSG\_DATABASE, 89  
MSG\_E-MAIL, 89  
MSG\_ERROR, 89  
MSG\_FILE, 89  
MSG\_FILE\_NAME, 89  
MSG\_FINISH, 89  
MSG\_NOTFOUND, 89  
MSG\_PASSWORD, 89  
MSG\_PROFILE, 89  
MSG\_RECIPIENT, 89  
MSG\_SQL, 89  
MSG\_START, 89  
Multiply, 208  
My Computer, 17

## N

NAME, 89  
New Type, 17  
NoDelay=1, 57  
non-metacharacters, 237, 281  
NotEqual, 193  
NotFileExists, 163  
Notification Actions, 37  
Notification Events, 37  
notification messages, 21  
Number, 238, 274  
Number of Retries, 34  
Number of Seconds to Wait, 28  
NUMBER\_OF\_RETRIES, 89  
Numeric functions, 270  
Numeric statements, 205

## O

Obtaining a number  
ODBC, 27, 67  
ODBC Interface, 29  
Off-line scripts, 99  
Operators, 289  
Options, 17, 25, 43, 44, 55, 57, 70, 94  
output file, 19  
Output files, 19

## P

Parameter Substitution, 38  
Password, 36  
PASSWORD, 89  
Paste, 75  
Paste JAL, 76  
Paste SQL, 76  
Pasting JAL Syntax, 76  
Pasting SQL Syntax, 76  
pattern, 282  
Ping, 181  
PingPort, 182  
Polling Interval, 19, 36  
POLLING\_INTERVAL, 89  
Port/Service, 56  
Power, 208  
Preparing to use your database, 66  
Print, 74, 210  
Print statements, 210  
Printer, 75  
PrinterGetDefault, 211  
PrinterPurgeAllJobs, 211  
PrinterSetDefault, 211  
Printing Scripts, 74  
Process, 44  
Process statements, 212  
ProcessGetExitCode, 220  
ProcessGetHandle, 213  
ProcessGetID, 213  
ProcessGetWindow, 215  
ProcessKill, 214  
ProcessList, 214  
Profile, 33  
PROFILE, 89  
ProfileInt, 268

ProfileString, 269  
Program Files and Documents, 17  
Program Name, 31  
Properties, 15, 50, 58  
Property Pages, 15  
Protecting/Unprotecting Job, 43  
Protection, 43  
proxy, 184, 185

## R

RaiseError, 103  
Rand, 275  
RAS statements, 224  
RASDial, 224  
RASGetStatus, 225  
RASHangUp, 225  
RawData=1, 57  
Reboot, 204  
REBOOT, 89  
Recommended Configuration, 26  
Redo, 75  
Refresh Rate, 80  
Registry statements, 226  
RegistryGetKey, 226  
RegistrySetKey, 227  
RelativeDate, 263  
RelativeTime, 263  
Remote Agent, 81  
Remote Agent Profiles, 62, 82  
Remote Agent Setup (Sample), 64  
Remote Agents, 60, 62, 63  
Remote Control, 81  
RemoteDir, 163, 169  
Rename, 41  
rename job, 41  
Repeat Every, 35  
Replace, 74, 239, 283  
Replace All, 74  
Replace With, 74  
Replacing in Scripts, 74  
Restart Windows, 32  
Restore, 12, 16  
Restoring network connection, 301  
Retry After Error, 34  
Retry Interval, 34  
RETRY\_INTERVAL, 89  
RETRY\_ON\_ERROR, 89  
Reverse, 240  
Right, 240, 284  
RightTrim, 284  
Round, 209, 275  
RTrim, 240  
Run Now, 43, 44  
RunAndWait, 217  
Running on network, 12  
RunWithInput, 218

## S

Saturday, 88  
SATURDAY, 89  
Save, 51  
Save Remote, 82  
Save to Remote Host, 82

SAVE\_ATTACHMENT, 89  
schedule a new task, 40  
SCHEDULE\_TYPE, 90  
Scheduling Jobs, 30  
Script, 33  
SCRIPT, 90  
Script Library, 98, 99  
Searching in Scripts, 73  
Second, 264  
SecondsAfter, 264  
Security issues, 12  
Semaphore File, 35  
Semaphore Files, 19  
Semaphore Process, 35  
Send e-mail, 37  
Send Keystroke, 33  
SEND\_KEYSTROKE, 90  
SendKeys, 221  
Service statements, 228  
ServiceContinue, 228  
ServiceGetStatus, 228  
ServicePause, 229  
ServiceStart, 230  
ServiceStop, 230  
Set, 110  
Setting, 132  
Settings, 13, 26, 61, 67  
Shortcut, 61, 83  
Sign, 276  
simultaneously, 81  
Skip, 20  
SKIP, 90  
Skip Holidays, 72  
Skip Job On Holiday, 35  
Skip late job, 33  
SKIP\_HOLIDAY, 90  
Slide Holidays, 72  
Slide Job On Holiday, 35  
SLIDE\_HOLIDAY, 90  
SMTP, 21  
Software Requirements, 26  
Space, 241, 285  
Special ASCII characters, 99  
SQL, 33, 90  
Sqrt, 276  
Standby, 58, 79  
Start, 13, 25, 67  
Start Date, 35  
Start In, 32  
Start Menu Programs, 13, 61  
Start Time, 35  
START\_DATE, 90  
START\_IN, 90  
START\_TIME, 90  
Starting 24x7 Remote Control, 83  
Starting Remote Agent, 61  
Startup, 45  
StartUp, 13, 61  
Statistics, 79  
Status Bar, 16  
Status Report, 70, 71  
Status report directory, 70  
Status report directory,, 70  
Stopping Job Execution, 45  
String, 241, 285

String functions, 277  
String statements, 231  
Subject, 36  
SUBJECT, 90  
Subtract, 209  
Sunday, 88  
SUNDAY, 90  
SyncFTPDir, 164  
synchronization interval, 55, 58  
SyncLocalDir, 166  
SyncRemoteDir, 167  
Syntax, 97  
Syntax for sort order, 126  
System Options, 94  
System Requirements, 26  
System Tray Icon, 16

## T

Taskbar, 13, 61  
Technical Support, 309  
Telnet statements, 243  
TelnetClose, 243  
TelnetConfig, 245  
TelnetOpen, 243  
TelnetReceive, 245  
TelnetSend, 244  
Testing Job Execution, 43  
Thursday, 88  
THURSDAY, 90  
Time, 135, 264  
Time Scale, 80  
TimeAdd, 136  
TimeDiff, 136  
Timeout, 32, 33  
TIMEOUT, 90  
Timer, 135  
To create Remote Control shortcut, 83  
Today, 137, 265  
Tools, 55, 58, 62, 70  
Tools/Options, 94  
Trace, 79  
Trace enabled, 55  
Trademarks, 13  
Transfer Mode for FTP, 170  
Translation, 151  
Translation Tables (EBCDIC/ASCII), 151  
Tree View, 50  
Trim, 242, 287  
Troubleshooting Job Execution, 47  
Truncate, 277  
Tuesday, 88  
TUESDAY, 90  
Two-digit years, 39

## U

Undo, 75  
Undo/Redo Changes, 75  
Uninstallation, 25  
Upper, 242, 288  
user-defined JAL statements, 98

Using Dynamic Data Exchange, 85  
Using FTP commands, 296  
Using JDL Files, 84  
Using the Editor, 73  
Using Windows messages, 300

## V

Value, 152  
Verifying overnight data feed, 307  
View, 17  
View menu, 79

## W

Wait, 223  
Watching for file changes, 302  
WebConfig, 183, 184, 185  
WebGetFile, 184  
WebGetPageHTML, 185  
WebOpenPage, 186  
WebPostData, 184  
Wednesday, 88  
WEDNESDAY, 90  
Weekday, 88  
Weekend, 88  
WINDOW, 90  
Window statements, 246  
Window Type, 32  
WindowActivate, 246  
WindowClickButton, 247  
WindowClose, 247  
WindowFind, 248  
WindowGetActive, 248  
WindowGetChild, 249  
WindowGetFirst, 249  
WindowGetLast, 250  
WindowGetNext, 251  
WindowGetParent, 251  
WindowGetPrevious, 252  
WindowGetProcess, 223, 252  
WindowGetTitle, 253  
WindowPostMessage, 255  
Windows 95  
Windows 95/98 specifics, 25  
Windows NT specifics, 24  
WindowSendMessage, 256  
WindowWaitClose, 253  
WindowWaitOpen, 254  
WinSock, 57  
WordCap, 288  
Working with Job Database, 21

## Y

Year, 265  
Yield, 204

## Z

Zoom, 54