

JP1 Version 11

**JP1/Script Description and Reference (For
Windows Systems)**

3021-3-B30(E)

Notices

■ Relevant program products

For details about the applicable OS versions, and the service packs and patches required for JP1/Script, see the *Release Notes*.

JP1/Script (For Windows):

P-2A12-3FBL JP1/Script version 11-00

■ Export restrictions

If you export this product, please check all restrictions (for example, Japan's Foreign Exchange and Foreign Trade Law, and USA export control laws and regulations), and carry out all required procedures.

If you require more information or clarification, please contact your Hitachi sales representative.

■ Trademarks

HITACHI, JP1, Job Management Partner 1, are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

Itanium is a trademark of Intel Corporation in the United States and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

MS-DOS is either a registered trademarks or a trademark of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Visual Basic is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Win32 is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows Server is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows Vista is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned in this document may be the trademarks of their respective owners.

Throughout this document Hitachi has attempted to distinguish trademarks from descriptive terms by writing the name with the capitalization used by the manufacturer, or by writing the name with initial capital letters. Hitachi cannot attest to the accuracy of this information. Use of a trademark in this document should not be regarded as affecting the validity of the trademark.

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Abbreviation		Full name or meaning
MS-DOS		Microsoft(R) MS-DOS(R) Operating System Version 5.0/V or later
Visual Basic		Microsoft(R) Visual Basic(R)
Win32		Microsoft(R) Win32(R)
Windows XP		Microsoft(R) Windows(R) XP Professional Operating System
Windows Server 2003	Windows Server 2003	Microsoft(R) Windows Server(R) 2003, Enterprise Edition
		Microsoft(R) Windows Server(R) 2003, Standard Edition
	Windows Server 2003 (x64)	Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition
		Microsoft(R) Windows Server(R) 2003, Standard x64 Edition
	Windows Server 2003 R2	Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition
		Microsoft(R) Windows Server(R) 2003 R2, Standard Edition
	Windows Server 2003 R2 (x64)	Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
		Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
Windows Vista	Windows Vista	Microsoft(R) Windows Vista(R) Business
		Microsoft(R) Windows Vista(R) Enterprise
		Microsoft(R) Windows Vista(R) Ultimate
	Windows Vista (x64)	Microsoft(R) Windows Vista(R) Business (x64)
		Microsoft(R) Windows Vista(R) Enterprise (x64)
		Microsoft(R) Windows Vista(R) Ultimate (x64)
Windows Server 2008	Windows Server 2008	Microsoft(R) Windows Server(R) 2008 Datacenter
		Microsoft(R) Windows Server(R) 2008 Enterprise
		Microsoft(R) Windows Server(R) 2008 Standard
	Windows Server 2008 (x64)	Microsoft(R) Windows Server(R) 2008 Datacenter (x64)
		Microsoft(R) Windows Server(R) 2008 Enterprise (x64)
		Microsoft(R) Windows Server(R) 2008 Standard (x64)
	Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Datacenter (x64)

Abbreviation		Full name or meaning
Windows Server 2008	Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Enterprise (x64)
		Microsoft(R) Windows Server(R) 2008 R2 Standard (x64)
Windows 7	Windows 7	Microsoft(R) Windows(R) 7 Professional
		Microsoft(R) Windows(R) 7 Enterprise
		Microsoft(R) Windows(R) 7 Ultimate
	Windows 7 (x64)	Microsoft(R) Windows(R) 7 Professional (x64)
		Microsoft(R) Windows(R) 7 Enterprise (x64)
		Microsoft(R) Windows(R) 7 Ultimate (x64)
Windows Server 2012	Windows Server 2012	Microsoft(R) Windows Server(R) 2012 Datacenter
		Microsoft(R) Windows Server(R) 2012 Standard
	Windows Server 2012 R2	Microsoft(R) Windows Server(R) 2012 R2 Datacenter
		Microsoft(R) Windows Server(R) 2012 R2 Standard
Windows 8	Windows 8	Windows(R) 8
		Windows(R) 8 Enterprise
		Windows(R) 8 Pro
	Windows 8 (x64)	Windows(R) 8 (x64)
		Windows(R) 8 Enterprise (x64)
		Windows(R) 8 Pro (x64)
Windows 8.1	Windows 8.1	Windows(R) 8.1
		Windows(R) 8.1 Enterprise
		Windows(R) 8.1 Pro
	Windows 8.1 (x64)	Windows(R) 8.1 (x64)
		Windows(R) 8.1 Enterprise (x64)
		Windows(R) 8.1 Pro (x64)
Windows 10	Windows 10 (x86)	Windows(R) 10 Enterprise 32-bit
		Windows(R) 10 Home 32-bit
		Windows(R) 10 Pro 32-bit
	Windows 10 (x64)	Windows(R) 10 Enterprise 64-bit
		Windows(R) 10 Home 64-bit
		Windows(R) 10 Pro 64-bit

Windows is sometimes used generically, referring to Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2012, Windows 8, Windows 8.1, and Windows 10.

■ Restrictions

Information in this document is subject to change without notice and does not represent a commitment on the part of Hitachi. The software described in this manual is furnished according to a license agreement with Hitachi. The license agreement contains all of the terms and conditions governing your use of the software and documentation, including all warranty rights, limitations of liability, and disclaimers of warranty.

Material contained in this document may describe Hitachi products not available or features not available in your country.

No part of this material may be reproduced in any form or by any means without permission in writing from the publisher.

■ Issued

Jan. 2016: 3021-3-B30(E)

■ Copyright

Copyright (C) 2016, Hitachi, Ltd.

Copyright (C) 2016, Hitachi Solutions, Ltd.

Preface

This manual describes the features and use of JP1/Script (P-2A12-3FBL).

■ Intended readers

This manual is intended for the following users:

- Users who have previously used a job control system
- Users who are familiar with the fundamentals of the operating system to be used, and who understand BASIC
- Users who are familiar with the functions of JP1 products

■ Organization of this manual

This manual is organized as follows:

PART 1. Description

1. Overview of JP1/Script

Chapter 1 provides an overview of JP1/Script functions and features, and describes the overall organization, types of programs, files, and system configuration of JP1/Script. It also explains the flow of operations in JP1/Script.

2. Preparation for Using JP1/Script

Chapter 2 explains how to install and uninstall JP1/Script, the accounts used for communication with other computers, and environment setup in various system environments. It also explains how to start and terminate JP1/Script.

3. JP1/Script Operations

Chapter 3 describes how to use JP1/Script Manager, Editor, Easy Input, Trace Viewer, Trace files, Menu Editor, Process Viewer, and Execution Environment File Converter. The windows that appear during JP1/Script operation are described in Chapter 4.

4. JP1/Script Dialog Boxes

Chapter 4 provides detailed descriptions of the dialog boxes that open from the Script Manager window, Script Editor window, Script Trace Viewer window, and Script Menu Editor window.

5. Troubleshooting

Chapter 5 describes the types of problems that may occur in JP1/Script and how to handle them.

PART 2. Reference

6. JP1/Script Coding Conventions

Chapter 6 describes the conventions relating to scripts used in JP1/Script and the conventions relating to command lines.

7. Statements

Chapter 7 describes the statements you can use when creating a script.

8. Basic Commands

Chapter 8 describes the basic commands you can use when creating a script.

9. Special Commands

Chapter 9 describes the special commands you can use when creating a script.

10. Script Control Interface

Chapter 10 describes the script control interface (API) for controlling a script during execution.

11. Script OLE Control

Chapter 11 describes the script OLE control for executing script commands.

A. Output Formats of Script Trace Files

Appendix A describes the output formats of the files that are handled by Script trace.

B. Sample Files

Appendix B summarizes the supplied sample files and explains their execution sequence.

C. Error Detail Codes

Appendix C describes the values and errors that are set in the reserved variable `__RTN__` when a JP1/Script command finishes executing.

D. Maintenance Log Files

Appendix D describes the maintenance log files that you must send when contacting the support center.

E. Version Changes

Appendix E describes the changes made in each JP1/Script version.

F. Reference Material for This Manual

Appendix F provides reference information on using this manual.

G. Glossary

Appendix G explains the terms used in JP1/Script.

Contents

Notices	2
Preface	6

Part 1: Description

1	Overview of JP1/Script	22
1.1	Features of JP1/Script	23
1.2	Overall organization of JP1/Script	24
1.3	JP1/Script component programs	25
1.4	Files used in JP1/Script	27
1.4.1	File types	27
1.4.2	File sizes	29
1.4.3	Large files	31
1.4.4	File access permissions	32
1.5	JP1/Script system configuration	35
1.5.1	System configuration linked with JP1/AJS	35
1.5.2	System configuration linked with JP1/Base	35
1.5.3	System configuration linked with JP1/IM	36
1.5.4	System configuration for using JP1/Script in a cluster environment	37
1.5.5	System configuration for using JP1/Script in a Remote Desktop service environment	37
1.6	JP1/Script operating procedure	39
1.7	Before creating and executing a script	40
1.7.1	About the Message command	40
1.7.2	About analysis and execution trace files	40
1.7.3	About the NetExec command	40
1.7.4	About the Beep command	40
1.7.5	About the restriction on paths for file names	41
1.7.6	About the Script Launcher service	41
1.8	Notes on executing JP1/Script	43
1.8.1	Starting and monitoring the JP1/Script service	43
1.8.2	Command behavior	44
1.8.3	How to set up Manager	47
1.8.4	How to change a registry entry	47
1.8.5	Message box layout	47
1.8.6	Execution as a user who has administrator permissions	47
1.9	About the lock error retry and trial-open functions	49
1.9.1	Lock error retry function	49

1.9.2	Trial-open function	52
2	Preparation for Using JP1/Script	54
2.1	Installing and uninstalling JP1/Script	55
2.1.1	Program installation folders	55
2.1.2	Installing JP1/Script	55
2.1.3	Registering the JP1/Script service	59
2.1.4	Uninstalling JP1/Script	59
2.1.5	Reinstalling JP1/Script	60
2.1.6	Upgrade installation of JP1/Script	61
2.1.7	Downgrade installation	62
2.2	Accounts for communication with other computers	63
2.3	Environment setup in a cluster system environment	64
2.3.1	Setting up an environment for using a Script Launcher	64
2.3.2	Setting up an environment for using a Script Launcher service	65
2.4	Environment setup in a Remote Desktop service environment	68
2.5	Renaming hosts and changing IP addresses	69
2.6	Monitoring of JP1/Script by the activity monitoring program	70
2.7	Starting and stopping JP1/Script	72
2.7.1	Starting JP1/Script	72
2.7.2	Stopping JP1/Script	74
2.8	Changing the system date and time	76
2.9	Setting users permitted to remotely execute the NetExec command	77

3 JP1/Script Operations 80

3.1	Manager operations	81
3.1.1	Script Manager window and menus	81
3.1.2	Mouse and key operations in the Script Manager window	85
3.1.3	Choosing an editor	86
3.1.4	Creating and saving a script	87
3.1.5	Creating a script with Easy Input	89
3.1.6	Editing a script	94
3.1.7	Checking script syntax	94
3.1.8	Copying a script	96
3.1.9	Adding a script	97
3.1.10	Deleting a script	99
3.1.11	Renaming a script	100
3.1.12	Setting the script execution environment (all items)	101
3.1.13	Setting the script execution environment (individual items)	104
3.1.14	Setting a script to start automatically	108
3.1.15	Executing a script	109
3.1.16	Creating a menu form	112

3.1.17	Starting and stopping Editor	114
3.2	Editor operations	115
3.2.1	Script Editor window and menus	116
3.2.2	Mouse and key operations in the Script Editor window	123
3.2.3	Setting and canceling comment lines	124
3.2.4	Using Easy Input	127
3.2.5	Creating a menu form	131
3.2.6	Setting the Editor operating environment	132
3.2.7	Checking script syntax	134
3.2.8	Starting and canceling monitoring	135
3.2.9	Executing a script	143
3.2.10	Setting and canceling breakpoints in monitoring mode	143
3.2.11	Setting the operating environment for monitoring mode	146
3.2.12	Adding a variable to the Watch window	147
3.2.13	Setting the script execution environment	148
3.2.14	Finding text in a file	151
3.2.15	Replacing text in a file	153
3.3	Easy Input operations	155
3.3.1	Starting Easy Input	155
3.3.2	Entering commands and statements	156
3.3.3	Specifying the JP1/Script version for script creation	157
3.3.4	Setting the line length and indent size	158
3.4	Trace Viewer operations	160
3.4.1	Script Trace Viewer window and menus	160
3.4.2	Mouse and key operations in the Script Trace Viewer window	163
3.4.3	Starting and stopping Trace Viewer	164
3.4.4	Operations in the Script Trace Viewer window	165
3.4.5	Showing or hiding trace files	165
3.4.6	Displaying trace file contents	166
3.4.7	Saving a trace file under a new name	167
3.4.8	Deleting a trace file	168
3.4.9	Clearing a trace file	170
3.4.10	Printing a trace file	171
3.4.11	Finding text in a trace file	172
3.5	Trace Files Display operations	174
3.5.1	Trace Files Display window and menus	174
3.5.2	Mouse and key operations in the Script Trace Files Display window	176
3.6	Menu Editor operations	178
3.6.1	Script Menu Editor window and menus	179
3.6.2	Mouse and key operations in the Script Menu Editor window	185
3.6.3	Creating a menu form	187

- 3.6.4 Copying a menu form 188
- 3.6.5 Pasting a menu form 189
- 3.6.6 Deleting a menu form 189
- 3.6.7 Changing control attributes in one operation 190
- 3.6.8 Aligning controls 193
- 3.6.9 Spacing controls equally 199
- 3.6.10 Centering controls on a menu form 201
- 3.6.11 Adjusting controls to the same size 203
- 3.6.12 Resizing a control to text size 206
- 3.6.13 Displaying a grid on a menu form 207
- 3.6.14 Setting the tab order 209
- 3.6.15 Moving a control to the foreground 211
- 3.6.16 Moving a control to the background 212
- 3.6.17 Defining the properties of a command 213
- 3.6.18 Displaying a menu form in test view 214
- 3.6.19 Printing a menu form 215
- 3.7 Process Viewer operations 216
- 3.7.1 Starting Process Viewer 216
- 3.7.2 Listing active script processes 218
- 3.7.3 Specifying a refresh interval for listing script processes 218
- 3.8 Execution Environment File Converter operations 220
- 3.8.1 Overview of Execution Environment File Converter 220
- 3.8.2 Converting an execution environment file 220
- 3.8.3 Executing Execution Environment File Converter 220
- 3.8.4 Overview of execution environment syntax files (.SPU) 223
- 3.8.5 Details of execution environment syntax files (.SPU) 223

4 JP1/Script Dialog Boxes 227

- 4.1 Script Manager dialog boxes 228
- 4.1.1 Create New Script File dialog box 228
- 4.1.2 Copy Files dialog box 230
- 4.1.3 Add Files dialog box 231
- 4.1.4 Confirm File Overwrite dialog box 231
- 4.1.5 Confirm File Deletion dialog box 232
- 4.1.6 Rename dialog box 233
- 4.1.7 Execution dialog box 234
- 4.1.8 Set Execution Environment (Start Information) dialog box 234
- 4.1.9 Set Execution Environment (Terminate Information) dialog box 237
- 4.1.10 Set Execution Environment (Trace Information) dialog box 238
- 4.1.11 Set Execution Environment (Command Line) dialog box 240
- 4.1.12 Set Command Line dialog box 241
- 4.1.13 Set Execution Environment (Working Folder) dialog box 242

- 4.1.14 Set Execution Environment (Start Date) dialog box 243
- 4.1.15 Set Execution Environment (Terminate Time) dialog box 245
- 4.1.16 Set Execution Environment (Trace Output Folder) dialog box 246
- 4.1.17 Set Execution Environment (User Trace Information) dialog box 247
- 4.1.18 Change Folder dialog box 248
- 4.1.19 File Properties dialog box 249
- 4.1.20 Set Automatic Startup dialog box 250
- 4.1.21 Link Editor dialog box 252
- 4.1.22 Options (Server Information) dialog box 252
- 4.1.23 Update Information dialog box 254
- 4.1.24 Options (Compatibility) dialog box 254
- 4.1.25 Options (Multi-activation) dialog box 255
- 4.1.26 Options (JP1/IM) dialog box 257
- 4.1.27 Options (Trace) dialog box 258
- 4.1.28 Options (Cluster Environment) dialog box 259
- 4.1.29 Select Folder dialog box 261
- 4.2 Script Editor dialog boxes 263
- 4.2.1 Update Value dialog box 263
- 4.2.2 Options (Format) dialog box 264
- 4.2.3 Options (Colors) dialog box 265
- 4.2.4 Options (Compatibility) dialog box 267
- 4.2.5 Set dialog box 267
- 4.2.6 Search dialog box 268
- 4.2.7 Set File Name dialog box 269
- 4.2.8 Set File Version dialog box 270
- 4.2.9 Command Line Parameter Settings dialog box 271
- 4.2.10 Add Variable dialog box 272
- 4.3 Script Trace Viewer dialog boxes 274
- 4.3.1 Confirmation (delete trace file) dialog box 274
- 4.3.2 Confirmation (clear trace file) dialog box 274
- 4.3.3 Select Computer dialog box 275
- 4.3.4 Search dialog box 276
- 4.4 Script Menu Editor dialog boxes 277
- 4.4.1 Menu Form Properties (General Items) dialog box 278
- 4.4.2 Menu Form Properties (Background) dialog box 279
- 4.4.3 Menu Form Properties (Key) dialog box 280
- 4.4.4 Menu Form Properties (Style) dialog box 282
- 4.4.5 Menu Form Properties (Wallpaper) dialog box 283
- 4.4.6 Static Properties (General Items) dialog box 284
- 4.4.7 Static Properties (Common Items) dialog box 285
- 4.4.8 Static Properties (Background) dialog box 287

4.4.9	Static Properties (Style) dialog box	288
4.4.10	Button Properties (General Items) dialog box	289
4.4.11	Button Properties (Common Items) dialog box	290
4.4.12	Button Properties (Background) dialog box	292
4.4.13	Button Properties (Key) dialog box	293
4.4.14	Button Properties (Style) dialog box	294
4.4.15	Browse Button Properties (General Items) dialog box	295
4.4.16	Browse Button Properties (Common Items) dialog box	296
4.4.17	Browse Button Properties (Background) dialog box	298
4.4.18	Browse Button Properties (Key) dialog box	299
4.4.19	Browse Button Properties (Style) dialog box	300
4.4.20	Edit Control Properties (General Items) dialog box	301
4.4.21	Edit Control Properties (Common Items) dialog box	303
4.4.22	Edit Control Properties (Key) dialog box	305
4.4.23	Edit Control Properties (Style) dialog box	306
4.4.24	Line Properties (General Items) dialog box	308
4.4.25	Function Key Properties (General Items) dialog box	309
4.4.26	Function Key Properties (Common Items) dialog box	310
4.4.27	Function Key Properties (Style) dialog box	312
4.4.28	List Control Properties (General Items) dialog box	313
4.4.29	List Control Properties (Common Items) dialog box	314
4.4.30	List Control Properties (Key) dialog box	316
4.4.31	List Control Properties (Style) dialog box	317
4.4.32	Combo Box Properties (General Items) dialog box	318
4.4.33	Combo Box Properties (Common Items) dialog box	319
4.4.34	Combo Box Properties (Key) dialog box	321
4.4.35	Combo Box Properties (Style) dialog box	322
4.4.36	Command Properties dialog box	324
4.4.37	Set Command Properties dialog box	325
4.4.38	Set Menu Form Name dialog box	330
4.4.39	Change as Batch dialog box	331
4.4.40	Set Grid dialog box	332
4.4.41	Set Tab Order dialog box	333
4.4.42	Print Information of Menu Forms dialog box	334

5 Troubleshooting 336

5.1	Troubleshooting procedure	337
5.2	Troubleshooting	338
5.2.1	Troubleshooting script execution problems	338
5.2.2	Troubleshooting command execution problems	339
5.2.3	Troubleshooting Trace Viewer problems	340
5.2.4	Troubleshooting Process Viewer problems	341

5.3	Log information	342
5.3.1	Types of log information	342
5.3.2	List of log files and directories	343
5.4	Information to be acquired in the event of an error	345
5.4.1	OS information	345
5.4.2	JP1/Script information	345
5.4.3	Details of operation	347
5.4.4	Information on the screen	347
5.5	How to acquire information	348
5.5.1	Data collection tool	348
5.5.2	Dump information	349
5.5.3	Acquiring JP1/Script information	349
5.5.4	Acquiring Windows event logs	349
5.6	Backup and recovery	350
5.6.1	Backing up and recovering the files used by JP1/Script	350
5.6.2	Backing up and recovering operating environment information	351

Part 2: Reference

6	JP1/Script Coding Conventions	352
6.1	Rules for creating scripts	353
6.1.1	Variable naming conventions	353
6.1.2	Maximum number and data size of variables	353
6.1.3	Restricted keywords	354
6.1.4	Reserved variables	355
6.1.5	Array variables	358
6.1.6	Constants	363
6.1.7	Numeric coding conventions	363
6.1.8	String coding conventions	363
6.1.9	Operation conventions	364
6.1.10	Operator precedence	364
6.1.11	Script coding conventions	364
6.1.12	JP1/Script exit codes	367
6.1.13	JP1/Script event logs	368
6.1.14	JP1 Events issued by JP1/Script	371
6.2	Rules for writing command lines	373
6.2.1	Command line formats	373
6.2.2	Command line parameters	374
6.2.3	Command line coding conventions	376
6.2.4	Notes on writing command lines	376

7 Statements 379

- 7.1 List of statements 380
 - 7.1.1 = 380
 - 7.1.2 Do...Loop 381
 - 7.1.3 For...Next 382
 - 7.1.4 For...End For 383
 - 7.1.5 If...Then...Else 384
 - 7.1.6 Select Case 386
 - 7.1.7 While...End 387
 - 7.1.8 Function 388
 - 7.1.9 Sub 389
 - 7.1.10 Call 391
 - 7.1.11 Exit xx 391
 - 7.1.12 GoTo 392
 - 7.1.13 Continue 393
 - 7.1.14 On Error 393

8 Basic Commands 395

- 8.1 List of basic commands 396
- 8.2 Commands for manipulating variables 402
 - 8.2.1 Dim (declare a variable and allocate space) 402
 - 8.2.2 Dim (array) (declare an array variable and allocate space) 402
 - 8.2.3 SetEnvironment or SetEnv (set an environment variable) 403
 - 8.2.4 GetEnvironment or GetEnv (get an environment variable) 405
 - 8.2.5 SetGV (set a global variable) 406
 - 8.2.6 GetGV (get a global variable) 407
 - 8.2.7 DeleteGV (delete a global variable) 409
 - 8.2.8 GetArrayCount (count the elements in an array variable) 410
- 8.3 Commands for manipulating strings 412
 - 8.3.1 InStr (find the position of a substring in a string) 412
 - 8.3.2 InArray (find the position of a string in an array variable) 413
 - 8.3.3 Len (calculate the length of a string) 414
 - 8.3.4 LCase (convert one-byte alphabetic characters to lowercase) 415
 - 8.3.5 UCase (convert one-byte alphabetic characters to uppercase) 415
 - 8.3.6 Left (return characters from the left side of a string) 416
 - 8.3.7 Mid (return characters from inside a string) 416
 - 8.3.8 Right (return characters from the right side of a string) 417
 - 8.3.9 Space (return a specified number of one-byte spaces) 418
 - 8.3.10 LTrim (remove leading spaces from a string) 418
 - 8.3.11 RTrim (remove trailing spaces from a string) 419
 - 8.3.12 Trim (remove leading and trailing spaces from a string) 420

8.3.13	+ operator (concatenate strings)	420
8.3.14	& operator (concatenate strings)	421
8.3.15	&= operator (concatenate strings)	422
8.3.16	AddStr (concatenate strings with delimiters inserted)	423
8.3.17	SeparateStrCount (count the number of separate strings)	424
8.3.18	SeparateStr (split a string into separate strings)	425
8.3.19	Str (convert a number to a string)	426
8.3.20	Format (convert a value to a formatted string)	427
8.3.21	IsLower (check whether a string is lowercase)	428
8.3.22	IsUpper (check whether a string is uppercase)	429
8.3.23	IsSingleChar (check whether a string is one-byte characters)	430
8.3.24	IsMultiChar (check whether a string is two-byte characters)	431
8.4	Commands for working with dates	432
8.4.1	Date (return the current date)	432
8.4.2	Time (return the current time)	432
8.4.3	Year (return the year for a specified date)	432
8.4.4	Month (return the month for a specified date)	433
8.4.5	Day (return the day for a specified date)	433
8.4.6	Weekday (return the weekday for a specified date)	434
8.4.7	Hour (return the hour for a specified time)	435
8.4.8	Minute (return the minute for a specified time)	435
8.4.9	Second (return the second for a specified time)	436
8.4.10	CalcDate (add and subtract dates)	436
8.4.11	CompDate (compare dates)	438
8.4.12	GetDateCount (calculate the difference between dates)	439
8.4.13	CalcTime (add and subtract times)	440
8.4.14	CompTime (compare times)	441
8.4.15	GetTimeCount (calculate the difference between times)	442
8.4.16	IsLeapYear (check whether a leap year)	443
8.5	Commands for managing files and folders	445
8.5.1	IniRead (read a value from an initialization file)	445
8.5.2	IniWrite (enter a value in an initialization file)	446
8.5.3	TextFileReplace (replace a string in a text file)	447
8.5.4	TextOpen (open a text file)	448
8.5.5	TextClose (close a text file)	450
8.5.6	TextRead (read one line of data from a text file)	451
8.5.7	TextWrite (write data to a text file)	452
8.5.8	TextSeek (move the read/write position to the file beginning or end)	453
8.5.9	GetTextPosition (return the current read/write position)	454
8.5.10	MakeDir (create a folder)	455
8.5.11	DeleteDir (delete a folder)	456

- 8.5.12 DeleteFile (delete a file) 457
- 8.5.13 Rename (rename a file) 458
- 8.5.14 TempDir (get the temporary folder name) 459
- 8.5.15 TempFile (create a temporary file) 460
- 8.5.16 SetFileAttribute or SetFileAttr (set folder or file attributes) 461
- 8.5.17 GetFileAttribute or GetFileAttr (get folder or file attributes) 462
- 8.5.18 SetFileTime (set file date and time) 463
- 8.5.19 GetFileTime (get file date and time) 464
- 8.5.20 GetFileSize (get file size) 465
- 8.5.21 GetVersionInfo or GetVerInfo (get version information for a file) 466
- 8.5.22 SplitFile (split a file) 468
- 8.5.23 CatFiles (join split files) 469
- 8.5.24 SetStandardFile or SetStdFile (set the standard input, output, or error file) 471
- 8.5.25 ResetStandardFile or ResetStdFile (reset the standard input, output, or error file) 473
- 8.5.26 SplitPath (analyze a full path) 474
- 8.5.27 MakePath (create a full path) 474
- 8.5.28 SetPath (set the path to the executable folder) 475
- 8.5.29 GetPath (get the path to the executable folder) 476
- 8.5.30 SetVolumeLabel or SetVolLabel (set a disk volume label) 477
- 8.5.31 GetVolumeLabel or GetVolLabel (get a disk volume label) 478
- 8.5.32 GetDiskFreeSpace (get the disk free space) 478
- 8.5.33 Copy (copy files) 479
- 8.6 Commands for message output 483
 - 8.6.1 InputBox (display a message and text boxes) 483
 - 8.6.2 Message (output text to a file or window) 484
 - 8.6.3 MessageBox (display a message in a dialog box) 488
 - 8.6.4 MessageEventLog (output a message to the application log in Event Viewer) 490
 - 8.6.5 IMEventMessage (issue events to JP1/IM or JP1/Base) 492
- 8.7 Command for menu display 495
 - 8.7.1 Menu (display a user-defined menu form) 495
- 8.8 Commands for performing calculations 497
 - 8.8.1 + operator (addition) (find the sum of two expressions) 497
 - 8.8.2 += operator (addition) (assign to a variable the sum of a variable and an expression) 498
 - 8.8.3 - operator (subtraction and negation) (find the difference between two numbers) 499
 - 8.8.4 -= operator (subtraction) (assign to a variable the difference between a variable and an expression) 499
 - 8.8.5 Mod operator (remainder calculation) (find the division remainder of two numbers) 500
 - 8.8.6 Mod= operator (remainder calculation) (assign to a variable the division remainder of a variable and an expression) 501
 - 8.8.7 * operator (multiplication) (find the product of two numbers) 501
 - 8.8.8 *= operator (multiplication) (assign to a variable the product of a variable and an expression) 502
 - 8.8.9 / operator (division) (find the quotient of two numbers) 503

- 8.8.10 /= operator (division) (assign to a variable the quotient of a variable and an expression) 503
- 8.8.11 \ operator (integer division) (find the quotient of two numbers) 504
- 8.8.12 \= operator (integer division) (assign to a variable the quotient of a variable and an expression) 505
- 8.8.13 ^ operator (power) (find the power of two numbers) 505
- 8.8.14 ^= operator (power) (assign to a variable a variable raised to the power of an expression) 506
- 8.8.15 Comparison operators (=, <>, <, <=, >, >=) (compare two expressions) 507
- 8.8.16 And operator (logical AND) (find the logical AND of two expressions) 508
- 8.8.17 Or operator (logical OR) (find the logical OR of two expressions) 509
- 8.8.18 Not operator (logical NOT) (find the logical NOT of an expression) 510
- 8.9 Commands for performing evaluations 511
 - 8.9.1 IsEmpty (check whether a variable is an Empty value) 511
 - 8.9.2 IsDefine or IsDef (check whether a variable is defined) 511
 - 8.9.3 IsNumeric (check whether a value is a numeric) 512
 - 8.9.4 IsEmptyDir (check whether a folder is empty) 512
 - 8.9.5 IsExistDir (check whether a folder exists) 513
 - 8.9.6 IsExistFile (check whether a file exists) 514
 - 8.9.7 IsWriteableDir (check whether a folder is writeable) 514
 - 8.9.8 IsFileAttribute or IsFileAttr (check a folder or file attribute) 515
 - 8.9.9 IsNew (compare files for the more recent version or date) 516
 - 8.9.10 CheckDirName (check whether a folder name ends with a backslash) 517
 - 8.9.11 CheckDriveType (check the drive type) 518
- 8.10 Commands for calling external programs 520
 - 8.10.1 Exec (call an executable file on the local PC) 520
 - 8.10.2 NetExec (call an executable file on the local PC or remote PC) 522
 - 8.10.3 WaitForExec (wait for completion or forcibly terminate an executable file) 526
 - 8.10.4 GetExecStatus (get the execution status of an executable file) 527
 - 8.10.5 CallSpt (call a script file with multiple parameters set) 528
- 8.11 Automatic start commands 533
 - 8.11.1 EntryStartUp (register a script file for automatic startup) 533
 - 8.11.2 CancelStartUp (cancel automatic startup for a script file) 534
- 8.12 Comment command 536
 - 8.12.1 Rem or ' (write a comment in a program) 536
- 8.13 Other commands 537
 - 8.13.1 Sleep (halt script execution) 537
 - 8.13.2 Alert (display or clear user error icons) 537
 - 8.13.3 Beep (sound a beep from the speakers) 538
 - 8.13.4 Exit (terminate script execution) 539
 - 8.13.5 GetErrorMessage (get an error message) 540
- 9 Special Commands 541**
 - 9.1 List of special commands 542
 - 9.2 Commands for registry operations 544

- 9.2.1 RegRead (read a value from the registry) 544
- 9.2.2 RegWrite (enter a value in the registry) 545
- 9.2.3 RegDelete (delete a value from the registry) 546
- 9.2.4 RegDeleteKey (delete a registry subkey) 547
- 9.3 Commands for displaying graphics 549
 - 9.3.1 BitmapShow (draw a bitmap) 549
 - 9.3.2 BitmapHide (erase a bitmap) 550
- 9.4 Commands for performing evaluations 551
 - 9.4.1 IsEmptyReg (check whether a registry subkey is empty) 551
 - 9.4.2 IsExistRegKey (check whether a registry subkey exists) 552
 - 9.4.3 IsExistService (check whether a service exists) 552
 - 9.4.4 IsEmptyGroup (check for shortcuts) 553
- 9.5 Commands for service operations 555
 - 9.5.1 ServiceSetValue (set service information) 555
 - 9.5.2 ServiceGetValue (get values in the service information) 557
 - 9.5.3 ServiceCreate (register a service) 559
 - 9.5.4 ServiceDelete (delete a service) 559
 - 9.5.5 ServiceStart (start a service) 560
 - 9.5.6 ServiceStop (stop a service) 561
 - 9.5.7 ServicePause (halt a service) 562
 - 9.5.8 ServiceContinue (resume a halted service) 563
 - 9.5.9 ServiceChange (change a setting for a service) 563
 - 9.5.10 ServiceQuery (get the information set for an active service) 564
 - 9.5.11 ServiceRefer (get the current status of a service) 565
 - 9.5.12 ServiceControl (send a control command to a service) 566
 - 9.5.13 GetServiceName (get the service name from a service display name) 567
- 9.6 Command for calling an external program 568
 - 9.6.1 CallDll (call a DLL file) 568
- 9.7 Commands for performing shortcuts 571
 - 9.7.1 MakeGroup (create a program group) 571
 - 9.7.2 DeleteGroup (delete a program group) 571
 - 9.7.3 MakeShortcut (create a shortcut) 572
 - 9.7.4 DeleteShortcut (delete a shortcut) 574
- 9.8 Commands for process monitoring 576
 - 9.8.1 GetProcessCount (get the number of process activations) 576
 - 9.8.2 GetProcessInfo (get process information) 577
 - 9.8.3 TerminateProcess (forcibly terminate a process) 578
- 9.9 Other commands 580
 - 9.9.1 ExitWindows (terminate a script and log off or shut down Windows) 580
 - 9.9.2 SetRetryMode (set the lock error retry function) 581
 - 9.9.3 ResetRetryMode (cancel the lock error retry function) 582

- 9.9.4 SetTrialOpenMode (set the trial-open function) 583
- 9.9.5 ResetTrialOpenMode (cancel the trial-open function) 584

10 Script Control Interface 585

- 10.1 About the script control interface 586
- 10.2 List of script control functions 587
 - 10.2.1 SPTHOpen (open the script control manager) 587
 - 10.2.2 SPTHClose (close the script control manager) 587
 - 10.2.3 SPTHGetErrorMessage (get an error message) 588
 - 10.2.4 SPTHTerminate (forcibly terminate script execution) 588
- 10.3 Coding example of a script control interface 591

11 Script OLE Control 593

- 11.1 About the script OLE control 594
- 11.2 SPTO control 595
 - 11.2.1 RTN property (return the error detail code for a method) 595
 - 11.2.2 EXECRTN property (return the exit code of an executable file) 595
 - 11.2.3 EXECID property (return the identifier of an executable file) 596
 - 11.2.4 EnableErrorMessage property (check for errors in the SPTO control) 596
 - 11.2.5 SetGV method (set a global variable) 597
 - 11.2.6 GetGV method (get a global variable) 597
 - 11.2.7 DeleteGV method (delete a global variable) 598
 - 11.2.8 Exec method (call an executable file) 598

Appendixes 600

- A Output Formats of Script Trace Files 601
 - A.1 Output formats of analysis trace files 601
 - A.2 Output formats of execution trace files 603
 - A.3 Output format of user trace files 606
 - A.4 Output format of server trace files 606
 - A.5 Output format of NetExec error log files 608
- B Sample Files 611
 - B.1 Sample file that executes processing according to numbers selected in a displayed menu 611
 - B.2 Sample file that calls an executable file and checks for a program completion code 612
- C Error Detail Codes 613
 - C.1 Error codes set by JP1/Script 613
 - C.2 OS error codes 613
- D Maintenance Log Files 618
- E Version Changes 621
 - E.1 Changes in version 11-00 621
 - E.2 Changes in version 10-00 621
 - E.3 Changes in version 09-00 621

F	Reference Material for This Manual	622
F.1	Related publications	622
F.2	Conventions: Abbreviations for product names	623
F.3	Online manuals	624
F.4	Conventions: Diagrams	624
F.5	Conventions: Fonts	624
F.6	Conventions: Symbols	625
F.7	About registry key names in 64-bit versions of Windows	625
F.8	About the ProgramData folder	625
F.9	Conventions: KB, MB, GB, and TB	626
G	Glossary	627

Index 633

1

Overview of JP1/Script

JP1/Script is a set of programs for creating and running scripts that control jobs in a Windows environment.

This chapter provides an overview of the JP1/Script functions and features, and describes the system configuration.

1.1 Features of JP1/Script

JP1/Script is a set of programs for creating and running scripts that control jobs on a Windows platform. You use JP1/Script to easily create scripts.

If you have experience performing job control on another system, you will find that you can use JP1/Script in the same manner to perform job control on a Windows system.

1. Easy program creation and execution

JP1/Script sequentially interprets and executes source code written using commands similar to BASIC. This means that JP1/Script requires no intermediate step between script creation or modification and execution, which simplifies program creation and execution.

2. Reduced workload during program editing and debugging

At the click of a button, JP1/Script's dedicated editor enables you to paste the commands and statements that you entered with the Easy Input facility directly into Editor without having to become conversant with the details of commands and statements. In addition, JP1/Script's monitoring facility can reduce the workload of debugging because it enables you to monitor execution of your scripts.

3. Commands for various purposes

JP1/Script provides the commands you will need for a wide variety of jobs, such as file and folder manipulation, message output, registry manipulation, and Windows shutdown.

4. Interactive program creation using menu functions

JP1/Script provides menu functions that enable you to run your script based on execution of window operations, thereby supporting interactive applications.

5. Wide variety of functions

JP1/Script provides a wide variety of functions.

For example, JP1/Script enables you to automatically start your script and execute one script from another script or start an application from a script on the local or on a remote computer.

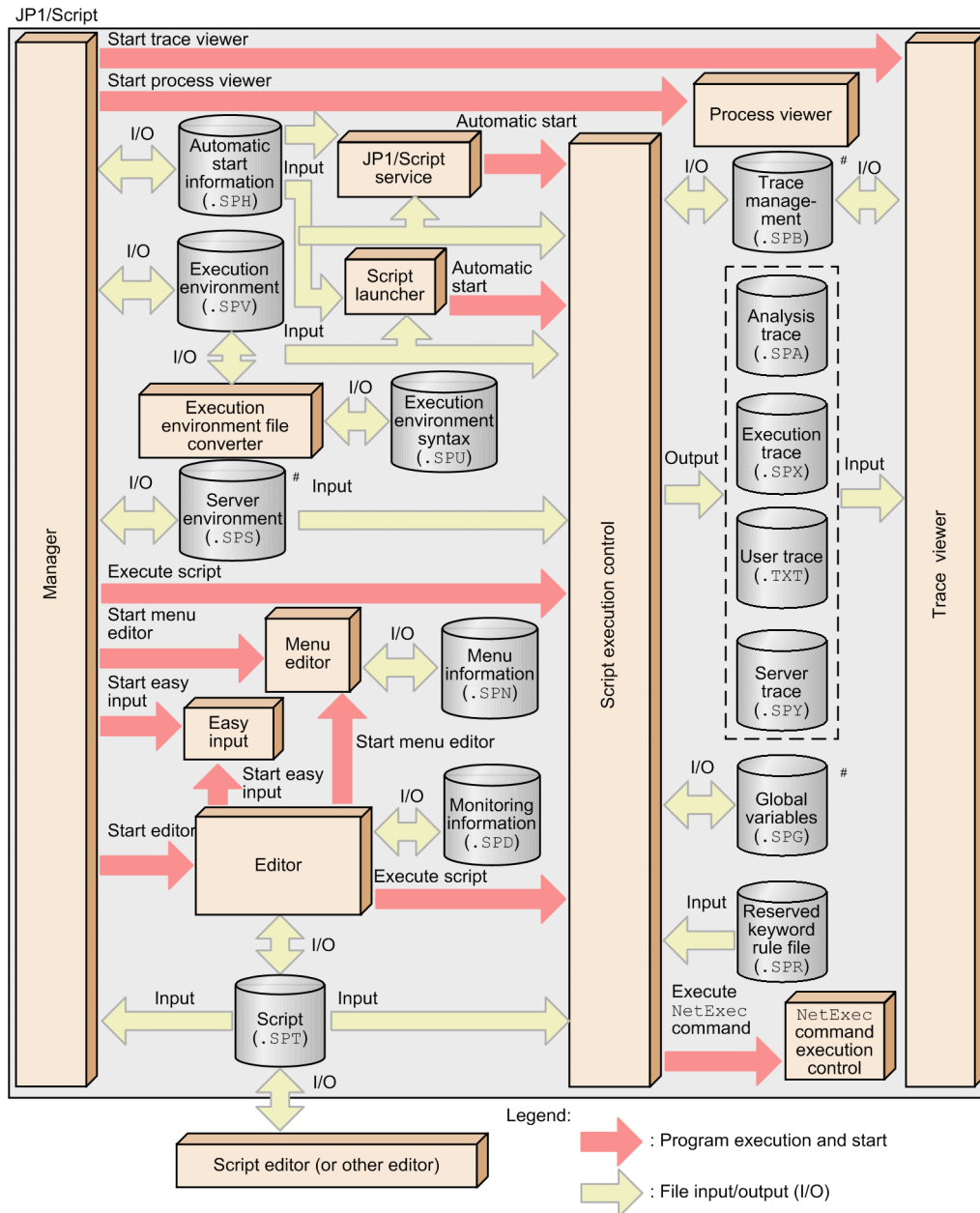
JP1/Script can also acquire program execution results. This enables you to determine the next process to be executed and achieve flexible linkage processing (such as program switchover).

When linked with other JP1 products, JP1/Script can achieve integrated job management.

1.2 Overall organization of JP1/Script

Figure 1-1 shows the overall organization of JP1/Script.

Figure 1-1: Overall organization of JP1/Script










#: Files for the JP1/Script service are input.

1.3 JP1/Script component programs

Table 1-1 lists the programs that make up JP1/Script. For a schematic that shows how the programs are interrelated, see [1.2 Overall organization of JP1/Script](#).

Table 1–1: JP1/Script component programs

Program	Icon	Program name	Overview	Necessity of process monitoring
Manager		SPTM.exe	Manages scripts created by JP1/Script and their execution environment.	N
Script execution control	--	Sptxe.exe	Analyzes and executes script files (.SPT).	N
NetExec command execution control	--	SPTXNetx.exe	Processes communication with the target computer during execution of NetExec command.	N
Script launcher		Spthlnch.exe	Controls start and termination of script files registered in the automatic start information in the logon space and NetExec command requests in the logon space.	Y
Script launcher service	--	SPTHLHSV.exe	Controls NetExec command requests in the logon space.	Y
JP1/Script service	--	SPTHSV.exe, SPTTMS.exe	Controls start and termination of script files registered in the automatic start information in the service space and NetExec command requests in the service space.	Y
Trace Viewer		SPTTM.exe	Displays the status and contents of trace files.	N
Easy Input		Sptke.exe	Simplifies input of commands and statements. This function enables the user to input commands without having to have detailed knowledge of the commands. The contents of the specified commands and statements can be pasted to the clipboard or an editor.	N
Editor		SPTIEdit.exe	JP1/Script's dedicated editor provides functions needed for creating, editing, and saving script files.	N
Menu Editor		SPTNEdit.exe	Provides functions for creating and editing the menu forms (windows) of created script files.	N
Process Viewer		SPTHView.exe	Provides functions for listing the script processes that are being executed and for monitoring and controlling processes.	N
Execution Environment File Converter	--	SPTUspv.exe	Mutually converts execution environment files (SPV extension) and execution environment syntax files (SPU extension).	N

Legend:

--: There is no icon for this program.

Y: Monitoring is required.

N: Monitoring is not required.

Information is also provided on the processes that are required in order to monitor the JP1/Script processes by an activity monitoring program, such as JP1/Cm2/SSO. For details about the monitoring of processes, see [2.6 Monitoring of JP1/Script by the activity monitoring program](#).

1.4 Files used in JP1/Script

1.4.1 File types

Table 1-2 describes the files that are used by JP1/Script.

For details about determining file size, see [1.4.2 File sizes](#). For details about the output formats of files that are handled by Trace Viewer, see [A. Output Formats of Script Trace Files](#).

Table 1–2: Files used by JP1/Script

File type	File name	File extension	File format	File contents
Files handled by Manager	Script file	.SPT	Text	File in which a created script is saved. You can specify any file name. You can specify no more than 30 characters. You cannot use spaces or the following symbols in file names: " = \ ; : . , { } < > / If you specify any of the above characters, operation cannot be guaranteed.
	Execution environment file	.SPV	Binary	Stores the environment for executing a script file. The file name is the same as the script file name.
	Automatic start information file	.SPH	Binary	Stores information for the Script launcher or JP1/Script service to execute scripts automatically. The fixed file name is SPTLNCH. Two types of files are provided, so that simultaneous logon by more than one user under the Fast User Switching feature is also supported: <ul style="list-style-type: none">• Automatic start information file for a specific user; applicable when Logon is set as the registered script's start type• Automatic start information file shared by all users; applicable when Service is set as the registered script's start type Both types have the fixed file name SPTLNCH.
	Server environment file	.SPS	Binary	Stores the server environment. The fixed file name is SPTS SV.
Files handled by Trace Viewer	Analysis trace file	.SPA	Text	Stores the results of analyzing script syntax. The file name is the same as the script file name.
	Execution trace file	.SPX	Text	Stores the results of executing the commands in a script. The file name is the same as the script file name.

File type	File name	File extension	File format	File contents
Files handled by Trace Viewer	User trace file (trace file)	.TXT	Text	Stores the trace produced at execution of the commands in a script. You can specify any file name.
	Server trace file	.SPY	Text	Stores the execution results of server commands called from commands written in a script running on a client. The fixed file name is SPTSVTRC.
Files handled by Editor	Monitoring information file	.SPD	Binary	Contains information used by the monitoring facility. The file name is the same as the script file name.
Files handled by Menu Editor	Menu information file	.SPN	Binary	Contains property definitions of a menu and controller. The file name depends on how Menu Editor was launched: <ul style="list-style-type: none"> When Menu Editor was launched from Manager or Editor: The file name is the same as the script file name. When Menu Editor was launched from the Windows Start menu: You can specify any file name.
Files handled by converter	Execution environment syntax file	.SPU	Text	Contains information used by Execution Environment File Converter. The information is set in the execution environment file in text format. You can specify any file name.
Other files	Trace management file ^{#1}	.SPB	Binary	Manages trace files. The fixed file name is SPTLOGDB.
	Work file ^{#2}	.TMP	Binary	Used internally by JP1/Script.
	Global variable file	.SPG	Binary	Stores the global variables set at script execution. The fixed file name is SPTGV.
	Reserved keyword rule file	.SPR	Binary	Stores the rules relating to the reserved words required for analyzing and executing script files. The fixed file name is VER $vvrr$, where vv is the JP1/Script version and rr is the JP1/Script revision.
	NetExec command restriction policy file	--	--	Stores the users who are permitted to execute the NetExec command. The field file names are SPTHSV_ACP and SPTHLSV_ACP.
	Maintenance log file	.LOG	Text	Records detailed information about the errors that occurred when Windows functions were called in JP1/Script programs.
	Program execution information management file	.CONF	Binary	Manages the program execution information file.

File type	File name	File extension	File format	File contents
Other files	Program execution information file	.LOG	Text	Records the execution status of the various JP1/Script programs.

#1

The trace management file manages such information as the names of the following trace files and the write start position in each trace file:

- Analysis trace file
- Execution trace file
- Server trace file
- User trace file

#2

JP1/Script temporarily creates work files in the `TEMP` folder. Note that if you make the `TEMP` folder subject to virus checks by antivirus software, JP1/Script applications might not operate normally.

Supplementary notes on the trace management file:

In the following cases, the size of the trace management file increases because the number of trace files that must be managed increases:

- A script is executed with different file names.
- `Target_File` is specified in the `Target` argument of the `Message` command to output to a new file.

When a script is executed with a different file name, the size of the trace management file increases in order to manage the analysis and execution trace files for that script file. When you execute the same script file more than once, the size of the trace management file does not increase because the command uses the existing information in the trace management file.

The size of the trace management file increases when data is output to a new file using the `Message` command with `Target_File` specified in the `Target` argument.

If the file specified in the `OutputName` argument of the `Message` command has already been output to the file by another `Message` command with `Target_File` specified in the `Target` argument, the command uses the information in the trace management file, and the file size does not increase.

When the `Message` command with `Target_File` specified in the `Target` argument is used to create many user trace files with unique file names, the size of the trace management file increases because the amount of information to be managed, such as file names and write start positions in the trace files, increases. When the size of the trace management file increases, the following events occur:

- The execution performance of script files becomes poor.
- Script file execution may terminate abnormally with termination code 20.
- Command execution may result in a memory shortage error.

When you create many user trace files with unique file names, use the `TextOpen`, `TextWrite`, and `TextClose` commands to create them. Because the files created by the `TextOpen`, `TextWrite`, and `TextClose` commands are not managed by the trace management file, these files do not increase the size of the trace management file.

1.4.2 File sizes

This section describes how to determine the size of each file used with JP1/Script and provides general guidelines about file size.

(1) Files handled by Manager

- Script file (.SPT)
The file size depends on the size of the script you create; set the appropriate value.
- Execution environment file (.SPV)
The file size is from 8 to 12 kilobytes.
- Automatic start information file (.SPH)
The file size depends on how many scripts you register for automatic startup. An initial allocation of 36 kilobytes is sufficient for seven scripts.
- Server environment file (.SPS)
The file size is from 8 to 12 kilobytes.

(2) Files handled by Trace Viewer

- Analysis trace file (.SPA)
Calculate the file size (in bytes) as follows:
$$\text{maximum-number-of-rows} \times (\text{maximum-number-of-columns} + 4)$$

The maximum numbers of rows and columns are set on the **User Trace Information** page, which is displayed by choosing Manager's **File** and then **Set Execution Environment**.
- Execution trace file (.SPX)
Calculate the file size (in bytes) as follows:
$$\text{maximum-number-of-rows} \times (\text{maximum-number-of-columns} + 4)$$

The maximum numbers of rows and columns are set on the **User Trace Information** page, which is displayed by choosing Manager's **File** and then **Set Execution Environment**.
- User trace file (.TXT)
Calculate the file size (in bytes) as follows:
$$\text{maximum-number-of-rows} \times (\text{maximum-number-of-columns} + 4)$$

The maximum numbers of rows and columns are set on the **User Trace Information** page, which is displayed by choosing Manager's **File** and then **Set Execution Environment**.
- Server trace file (.SPY)
The file size is from 0.2 to 204 kilobytes.

(3) Files handled by Editor

- Monitoring information file (.SPD)
The file size depends on the number of watch variables and breakpoints. The initial allocation is 4 kilobytes.

(4) Files handled by Menu Editor

- Menu information file (.SPN)
The file size depends on how many forms you define and how many controls are pasted on each form. Set an appropriate value.

(5) Files handled by the converter

- Execution environment syntax file (.SPU)

The file size depends on the size of the execution environment file (.SPV).

(6) Other files

- Trace management file (.SPB)
The minimum and secondary allocations are each 520 kilobytes. The secondary allocation per trace file is from 0.3 to 1 kilobyte. The initial allocation size can accommodate 8 trace files.
- Global variable file (.SPG)
The initial and secondary allocations are each 20 kilobytes.
- Maintenance log file (.LOG)
The maximum file size when the file is output with the default value is 4,032 kilobytes.
- Program execution information management file (.CONF)
The maximum file size allowed is 1 kilobyte.
- Program execution information file (.LOG)
The maximum file size allowed is 20,480 kilobytes.

1.4.3 Large files

(1) File system

JP1/Script supports large files for FAT32 and NTFS systems (neither FAT nor FAT16 is supported).

(2) Maximum file size

The maximum file size is 8 exabytes (9,223,372,036,854,775,807 bytes) for NTFS and 3.9 gigabytes (4,294,967,295 bytes) for FAT32. If the size of a file exceeds this value, operations cannot be guaranteed.

(3) Error messages

The following table describes the error messages for large files:

No.	Command	Error message	Reserved variable (value)
1	GetFileSize	Value cannot be stored in the variable because the acquired value is greater than the variable's maximum value.	<code>_ERR_FILE_SIZE_</code> (536904784)
2	IniRead	Specified file size exceeds the maximum permissible size.	<code>_ERR_NOT_LARGE_FILE_</code> (536904785)
3	IniWrite	Specified file size exceeds the maximum permissible size.	<code>_ERR_NOT_LARGE_FILE_</code> (536904785)
4	TextFileReplace	Specified file size exceeds the maximum permissible size.	<code>_ERR_NOT_LARGE_FILE_</code> (536904785)
5	GetTextPosition	Read/write start location is beyond 2,147,483,647.	<code>_ERR_FILE_POSITION_</code> (536904786)

1.4.4 File access permissions

You can configure access permissions for files created by JP1/Script according to your operational needs. By setting appropriate access permissions for each type of file, you can prevent security risks. You can set the following access permissions:

- Access permissions are not set. (No permissions are specified.)
- Inherit access permissions from parent folders
- Everyone: Full control

To set access permissions, set a value for the following registry key:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option

Value name

SecurityAttributesSucceed

Value datatype

REG_DWORD

Value

- 0: Access permissions are not set.
 - 1: Access permissions are set to "Inherit access permissions from parent folders".
 - 2: Access permissions are set to "Everyone: Full control".
- If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

The following table lists the files that can be created in JP1/Script and the access permissions that can be set.

Table 1–3: Files for which access permissions can be set

File name	Extension	Access permissions	
		Inherit access permissions from parent folders	Everyone
Script file [#]	.SPT	Y	Y
Execution environment file	.SPV	Y	Y
Automatic start information file	.SPH	Y	N
Server environment file	.SPS	Y	N
Analysis trace file	.SPA	Y	Y
Execution trace file	.SPX	Y	Y
User trace file	.TXT	Y	Y
Server trace file	.SPY	Y	Y
Monitoring information file	.SPD	Y	Y
Menu information file	.SPN	Y	Y
Trace management file	.SPB	Y	N

File name	Extension	Access permissions	
		Inherit access permissions from parent folders	Everyone
Global variable file	.SPG	Y	N

Legend:

Y: Can be set

N: Cannot be set

#

Access permissions for script files that are copied or added by using Script Manager inherit the access permissions from parent folders, regardless of the values set in the registry.

The following table lists the access permissions that can be set when a new file is created by using an operation command.

Table 1–4: Access permissions that can be set for created files

Command	Access permissions		Remarks
	Inherit access permissions from parent folders	Everyone	
IniRead	N	N	Access permissions are not changed.
IniWrite	N	N	When new files are created, access permissions are inherited from parent folders.
TextFileReplace	N	N	Access permissions are not changed.
TextOpen	Y	Y	Access permissions are not changed for existing files.
TextClose	N	N	Access permissions are not changed.
TextRead	N	N	Access permissions are not changed.
TextWrite	N	N	Access permissions are not changed.
TextSeek	N	N	Access permissions are not changed.
GetTextPosition	N	N	Access permissions are not changed.
MakeDir	N	N	Access permissions are inherited from parent folders.
Rename	N	N	Access permissions are not changed.
TempDir	N	N	Access permissions are not changed.
TempFile	N	N	Access permissions are inherited from parent folders.
SplitFile [#]	Y	Y	Access permissions are not changed for existing files.
CatFiles [#]	Y	Y	Access permissions are not changed for existing files.
SetStandardFile	Y	Y	Access permissions are not changed for existing files.
Copy [#]	Y	Y	If <code>Security</code> is specified for the values of <code>Option6</code> and <code>Option7</code> , access permissions for the source files are set.
Message [#]	Y	Y	Access permissions are not changed for existing files.
MakeGroup	N	N	Access permissions are inherited from parent folders.

Legend:

Y: Can be set

N: Cannot be set

#

When you create a file by using the `SplitFile`, `CatFiles`, `Copy`, or `Message` command, if the folder in which the file is to be created does not exist, the folder is created. The access permissions for the created folder are inherited from parent folders.

1.5 JP1/Script system configuration

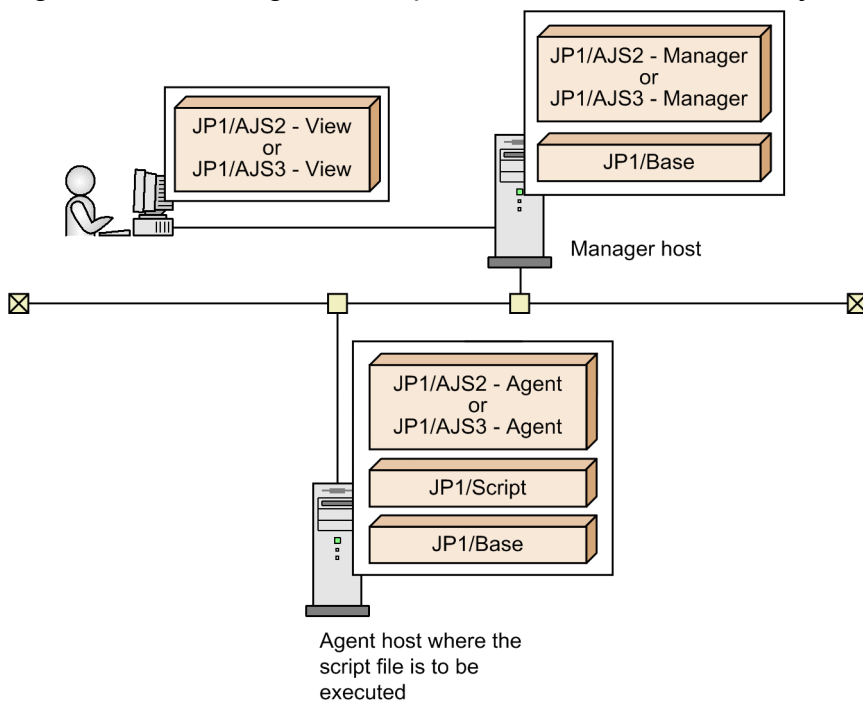
This section describes the system configuration using JP1/Script.

1.5.1 System configuration linked with JP1/AJS

This subsection describes the system configuration for executing and managing jobs automatically by linking JP1/Script with JP1/AJS (JP1/AJS2 or JP1/AJS3). JP1/AJS enables you to define and execute as a job a script file created by JP1/Script.

Figure 1-2 shows how to link JP1/Script with JP1/AJS to achieve automatic execution and management of jobs.

Figure 1–2: Linking JP1/Script with JP1/AJS to achieve job execution and management



If you create in the Manager host's job definition a job for issuing a request to execute a script file created by an agent host, the agent host executes the script file at the specified execution time.

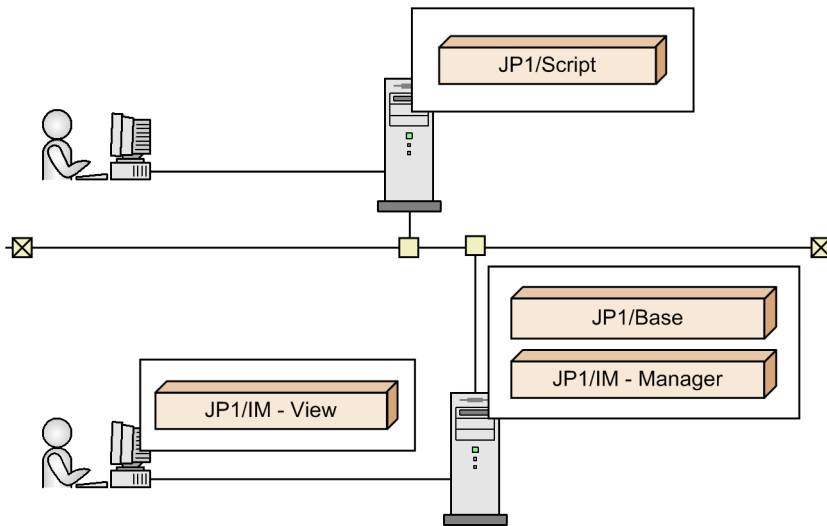
JP1/AJS does not support execution of a program that displays GUI and then waits for entry of the job. JP1/Script enables you to use a script file to start a program with GUI. For details, see [8.10.2 NetExec \(call an executable file on the local PC or remote PC\)](#).

1.5.2 System configuration linked with JP1/Base

This subsection describes the system configuration for linking JP1/Script with JP1/Base and issuing JP1 events from JP1/Script to JP1/Base. JP1/Script can issue JP1 events to JP1/Base. For details about issuing JP1 events, see [4.1.26 Options \(JP1/IM\) dialog box](#) and [8.6.5 IMEventMessage \(issue events to JP1/IM or JP1/Base\)](#).

Figure 1-3 shows how to link the JP1/Script system configuration with JP1/Base.

Figure 1–3: Linking JP1/Script with JP1/Base

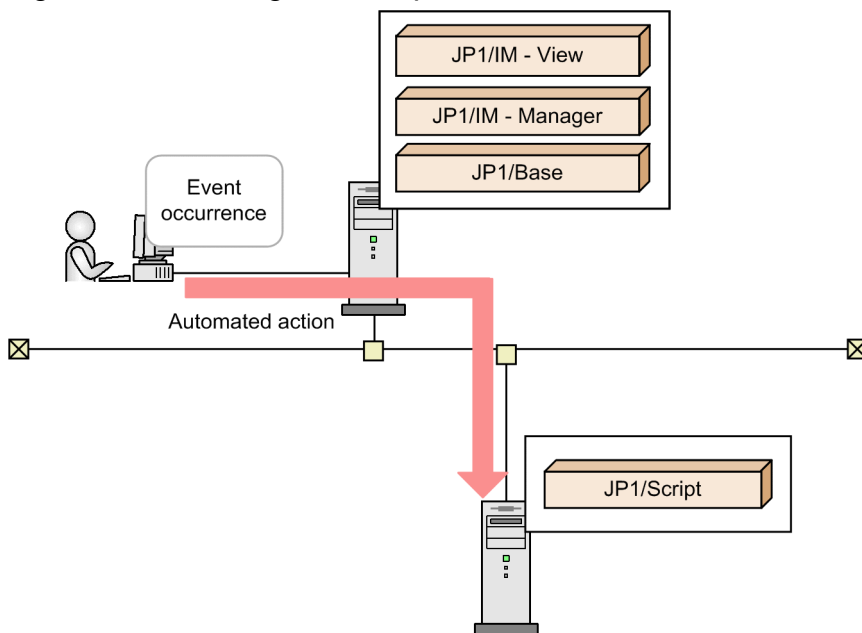


1.5.3 System configuration linked with JP1/IM

This subsection describes the system configuration for linking JP1/Script with JP1/IM. JP1/IM enables you to execute a script file created by JP1/Script by means of an automated action, which means that commands are executed automatically when a specific JP1 event is received.

Figure 1-4 shows how to link the JP1/Script system configuration with JP1/IM.

Figure 1–4: Linking JP1/Script with JP1/IM

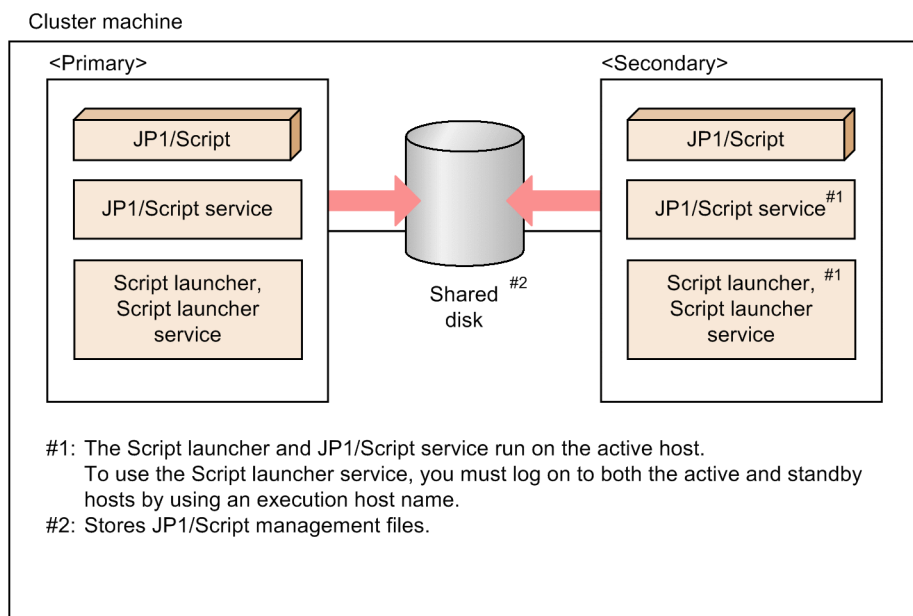


1.5.4 System configuration for using JP1/Script in a cluster environment

This subsection describes the system configuration for using JP1/Script in a cluster environment. JP1/Script supports a cluster environment, but does not inherit execution status (such as failover). For details about the cluster environment, see [2.3 Environment setup in a cluster system environment](#).

Figure 1-5 shows the system configuration for using JP1/Script in a cluster environment.

Figure 1–5: System configuration for using JP1/Script in a cluster environment

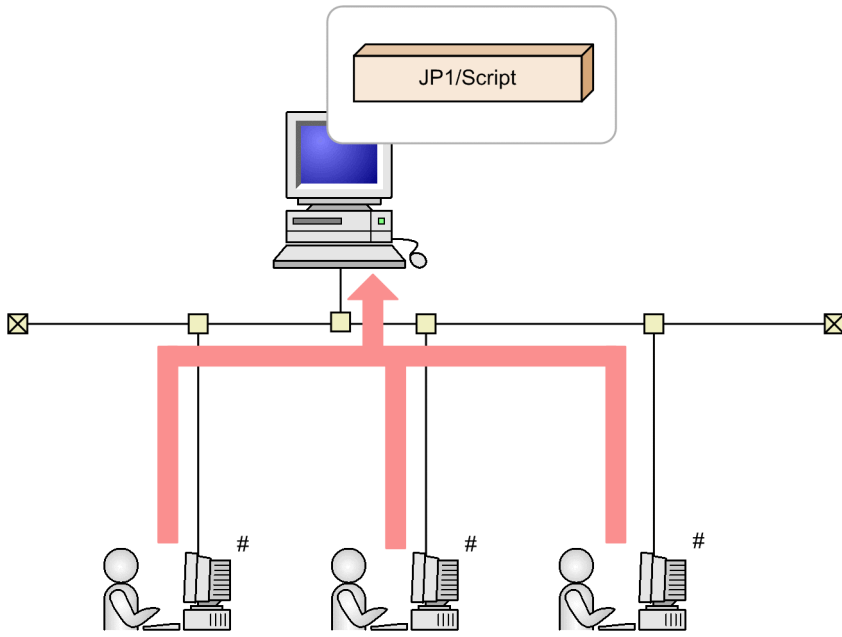


1.5.5 System configuration for using JP1/Script in a Remote Desktop service environment

This subsection describes the system configuration for using JP1/Script in a Remote Desktop service environment. For details about the Remote Desktop service environment, see [2.4 Environment setup in a Remote Desktop service environment](#).

The following figure shows the system configuration for using JP1/Script in a Remote Desktop service environment.

Figure 1–6: System configuration for using JP1/Script in a Remote Desktop service environment



You need to buy the P-L112-3GBL JP1/Script - Access License.

1.6 JP1/Script operating procedure

JP1/Script operations can be divided broadly into two types, operations in the Manager window and operations in the Trace Viewer window (and View Trace File window).

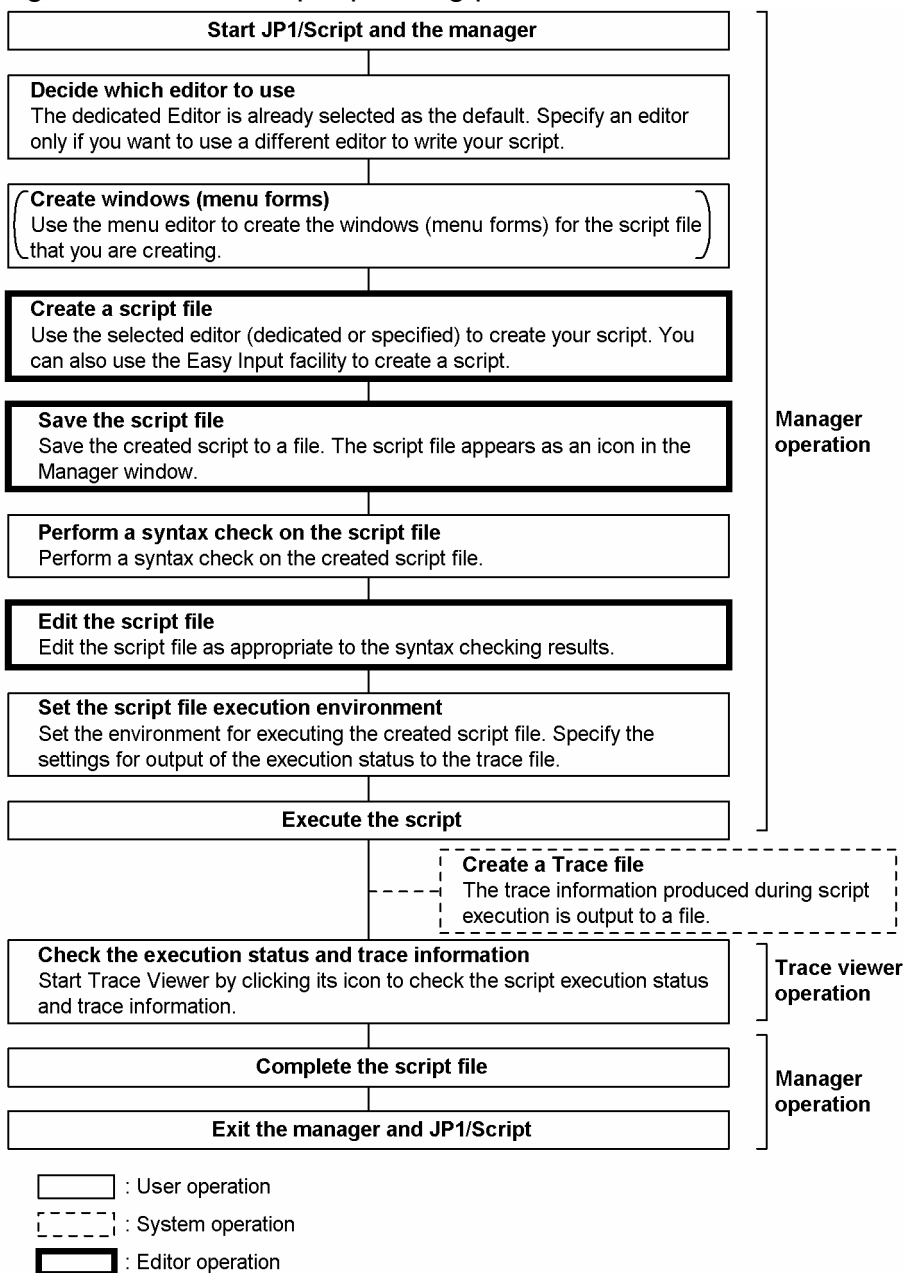
You use the Manager window to perform a series of operations for completing a script. When you actually code the script, you must start Editor (or a user-specified editor) from the Manager window.

You use the Trace Viewer window to manipulate the trace file that is created by the script.

This section describes the procedure up to the point of completion of a script file in the Manager window.

To create a script file, start Editor from Manager. You can use the syntax checker to perform lexical and grammatical analysis of the script.

Figure 1–7: JP1/Script operating procedure



1.7 Before creating and executing a script

This section describes notes before creating and executing a script.

1.7.1 About the Message command

When a `Message` command with `Target_File` specified in the `Target` argument is used to create many user trace files with unique file names, the size of the trace management file increases because the amount of information to be managed, such as file names and write start positions in the trace files, increases. When the size of the trace management file increases, the following events occur:

- The execution performance of script files becomes poor.
- Script file execution may terminate abnormally with termination code 20.
- Command execution may result in a memory shortage error.

When you create many user trace files with unique file names, use the `TextOpen`, `TextWrite`, and `TextClose` commands to create them. Because the files created by the `TextOpen`, `TextWrite`, and `TextClose` commands are not managed by the trace management file, these files do not increase the size of the trace management file.

1.7.2 About analysis and execution trace files

The default is that the analysis and execution trace files output error information that is generated during script execution. You can set the execution environment so that no trace files are output. However, if you do so you may not be able to identify the causes of errors, because the statements and the command errors that may occur during script execution are not output. It is recommended that you output trace files if at all possible. However, if you specify output of a detailed execution trace file[#], execution performance of script files will be adversely affected because the amount of output information will increase.

If you choose not to output trace files, you should create scripts that take into account error handling, such as by checking each command's execution results and identifying the causes of errors by saving to a file the names of erroneous commands and the contents of `_RTN_` reserved variables.

[#]: For details about the output levels of the execution trace file, see [4.1.12 Set Command Line dialog box](#) or [6.2.2\(3\) / SPXLV\(n\) \(or /spxlv\(n\)\)](#).

1.7.3 About the NetExec command

If multiple scripts that execute the `NetExec` command are executed, script execution performance is adversely affected, because processing with the target computer occurs for each `NetExec` command that is executed. You should not execute a large number of scripts that execute the `NetExec` command.

1.7.4 About the Beep command

Note that the `Beep` command might not sound a beep depending on the OS and hardware environment in which the command is executed. For details, see the note in [8.13.3 Beep \(sound a beep from the speakers\)](#).

1.7.5 About the restriction on paths for file names

You can use no more than 258 bytes to specify the absolute path of a JP1/Script file, such as an execution file or text file. To specify a relative path, you can use no more than 258 bytes for the path from the top-level directory in the unit to the target file.

1.7.6 About the Script Launcher service

The Script Launcher service allows you to control `NetExec` command requests in the logon space in remote sessions. To operate a Remote Desktop Session Host server in a Remote Desktop service environment, you must log on from a remote session. However, because Script Launcher is started in the primary session on the Remote Desktop Session Host server and cannot be started in a remote session, you cannot use Script Launcher to control `NetExec` command requests in the logon space. However, you can control `NetExec` command requests in the logon space in such a system configuration by using the Script Launcher service.

The Script Launcher service causes the programs that will be started by the `NetExec` command with the logon space specified to be started in the logon session of the user specified for the start parameter of this service. Therefore, to use the `NetExec` command while the Script Launcher service is running, you must stay logged on. You can log on either from the console or from a Remote Desktop service. Normally, you do not have to stay logged on, provided that you are logged on when using the `NetExec` command. If you log on through the Remote Desktop service, you will stay logged on after disconnecting the session.

The Script Launcher service is registered in the Windows service with the following information:

- General
 - Service name: `JP1_SCRIPT_LAUNCHER`
 - Display name: JP1/Script Launcher Service
 - Description: None
 - Executable file path: *installation-folder*\Bin\SPTHLHSV.EXE
 - Startup type: Manual
- Logon
 - Local system account
 - Only the local system account can be set as the start account for the Script Launcher service.
- Recovery
 - First error: Nothing is performed.
 - Next error: Nothing is performed.
 - Subsequent errors: Nothing is performed.
 - Error count reset: 0
- Dependencies
 - System components on which the service depends: None
 - System components that depend on the service: None

When you use the Script Launcher service, Script Launcher can no longer be started. This means that automatic startup in the logon space is no longer possible. If you want automatic startup in the logon space, use Script Launcher rather than the Script Launcher service.

To use the Script Launcher service:

1. Delete the Script Launcher registration in the startup menu.
2. Use Windows Service Manager to change the properties of the Script Launcher service as shown below. Then, with the Properties dialog box open, start the Script Launcher service.

On the **General** page:

- Startup type: Automatic
- Start parameters: Logon user name

For the logon user name, specify the logon space account name used for executing the command requested by the `NetExec` command.

Specify the logon user name in *user-name*, *user-name@domain-name*, or *domain-name\user-name* format.

Do not close the Properties dialog box before the Script Launcher service has started. If you do so, the information set for the start parameters will be lost. If the information is lost, set the information on the **General** page again, and then start the Script Launcher service.

3. Log on with the logon user name specified for the start parameter.

Do not log off while you are using the `NetExec` command. If you have logged on through the Remote Desktop service, you can terminate the Remote Desktop Services client if you stay logged on after disconnecting the session.

To change the logon space account name used for executing the commands requested by the `NetExec` command, stop the Script Launcher service and then start from step 2.

To specify a user other than an administrator for the logon space account name, assign the **Create global objects** permission in the Windows security settings in advance.

1.8 Notes on executing JP1/Script

This section describes notes on executing JP1/Script.

1.8.1 Starting and monitoring the JP1/Script service

(1) When to start the JP1/Script service

The JP1/Script service must always be running. Before you attempt to start a script file from another service, you need to start the JP1/Script service.

Adverse effects if the JP1/Script service is not running

The system is affected as follows if the JP1/Script service is not running:

- An analysis or execution trace is not output when a script file is started.
If you start a script file while the JP1/Script service is not running, the following warning message is output to the event log:
ID 96: The execution result will not be output to the analysis/execution trace file because the JP1/Script service is not running.\nAlso, execution of the Message, EntryStartUp, or CancelStartUp command, or the global variable operation command will result in an error.\nStart the JP1/Script service, and then perform execution.
Note that startup of a script file will not be canceled even if the JP1/Script service is not running.
If the JP1/Script service starts running during execution of the script file, the subsequent analysis and execution traces will be output.
- Execution of the Message command with Target_File, Target_SPXFile, or Target_SPAFile specified, the EntryStartUp command, the CancelStartUp command, or a command for manipulating variables fails.
If an error occurs while the JP1/Script service is not running, `__ERR_SERVICE_NOT_BEGIN__` (value: 536903968) is set in the `__RTN__` reserved variable.
- The `TerminateProcess` command cannot forcibly terminate a script process that was executed by a service or another user.
The command always returns `True`. This is the same as what happens when a process specified by the process ID does not exist.
- In the Script Manager window, automatic start information cannot be set in the dialog box that opens by selecting: (1) **File, Set Execution Environment**, and then **All Items**, (2) **File, Set Execution Environment**, and then **Start Date**, or (3) **Tools** and then **Set Automatic Start**.
If you perform the operation while the JP1/Script service is not running, the following error dialog box opens and the display of the setup dialog box is canceled:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.
- Options cannot be set in the dialog box that opens by selecting **Tools** and then **Options** in the Script Manager window.
If you perform the operation while the JP1/Script service is not running, the following error dialog box opens and the display of the option dialog box is canceled:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

- The dialog box for setting automatic start information cannot be displayed by selecting **Monitoring** and then **Set Execution Environment** in the Script Editor window.

Automatic start information cannot be set. If you perform the operation while the JP1/Script service is not running, the following error dialog box opens and the display of the setup dialog box is canceled:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

- Processes being executed by the service and other users cannot be displayed in Process Viewer.

If the processes cannot be displayed, the following error dialog box opens, and then Process Viewer starts:

Processes that are started by a service or being executed by another user are not displayed because the JP1/Script service is not running. To display the processes, launch the JP1/Script service, and then re-execute the operation.

- Trace Viewer cannot be started.

If Trace Viewer cannot be started, the following error dialog box opens and startup of Trace Viewer is canceled:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(2) Monitoring the JP1/Script service

If necessary, use the activity monitoring program to monitor the JP1/Script service. For details, see [2.6 Monitoring of JP1/Script by the activity monitoring program](#).

1.8.2 Command behavior

If you execute a script file from a service, operation is performed with the user account permissions specified for that service. However, if you execute a script file from Manager, JP1/Script applications run as if executed by a standard user due to User Account Control (UAC) for Windows. This is true even if you have logged on as a member of the Administrators group.

The following commands cause problems when they are executed by a standard user :

- Commands that might cause an Access Denied error
- File operation commands
- Exec and NetExec commands
- TerminateProcess command
- GetProcessInfo command

(1) Commands that cause an Access Denied error

Table 1-3 lists the commands that might cause an Access Denied error.

You must execute script files that execute these commands as a user who has administrator permissions. If you execute one of these commands as a standard user, the following error code is output:

0005: Access was denied. Recheck file attributes or security.

For details about how to execute script files as a user who has administrator permissions, see [1.8.6 Execution as a user who has administrator permissions](#).

Table 1–5: TableCommands that might cause Access Denied error

No.	Command	Condition on which an Access Denied error occurs
1	SetEnvironment	SystemEnv is specified for the first argument (Type).
2	Rename	Reboot is specified for the third argument (Method).
3	SetVolLabel	None
4	RegWrite	A value other than HKEY_CURRENT_USER is specified for the first argument (RegKey). ^{#1}
5	RegDelete	A value other than HKEY_CURRENT_USER is specified for the first argument (RegKey). ^{#2}
6	RegDeleteKey	A value other than HKEY_CURRENT_USER is specified for the first argument (RegKey). ^{#3}
7	ServiceCreate	None
8	ServiceDelete	None
9	ServiceStart	None
10	ServiceStop	None
11	ServicePause	None
12	ServiceContinue	None
13	ServiceChange	None
14	ServiceControl	None
15	MakeGroup	Lcl_Program is specified for the second argument (RootType).
16	DeleteGroup	Lcl_Program is specified for the second argument (RootType).
17	MakeShortcut	Lcl_Desktop, Lcl_Startmenu, Lcl_Program, or Lcl_Startup is specified for the first argument (RootPath)
18	DeleteShortcut	Lcl_Desktop, Lcl_Startmenu, Lcl_Program, or Lcl_Startup is specified for the first argument (RootPath).
19	GetProcessCount	None
20	GetProcessInfo	None
21	TerminateProcess	None
22	Copy	Security is specified for the ninth (Option6) or tenth (Option7) argument.

#1

If you specify HKEY_LOCAL_MACHINE for the first argument (RegKey) and specify Software for the first key of the second argument (SubKey), no error occurs because the command redirects the entry to HKEY_CURRENT_USER\Software\Classes\VirtualStore\Machine\Software. The value in the entry redirected by the RegWrite command will be preferentially read by the RegRead command.

#2

To delete the redirected value from the Software folder, you need to specify the redirect destination key.

#3

If you execute the command for the redirected key in the Software folder, the key is not deleted.

(2) File operation commands

Note that if you use a file operation command to output or update files in the folder specified by the `ProgramFiles` or `WinDir` environment variable, the files might be redirected to the area allocated for the user. The redirect destination is in the `VirtualStore` folder whose path is specified in the `LocalAppData` environment variable. The files are redirected if a script file that executes file operation commands is executed by a standard user. To prevent the files from being redirected, you need to execute the script file as a user who has administrator permissions. For details about how to execute script files as a user who has administrator permissions, see [1.8.6 Execution as a user who has administrator permissions](#).

If a standard user uses script files to execute file operation commands to input or view data, operations are preferentially performed on the files at the redirect destination.

(3) Exec and NetExec commands

If a standard user executes a script file that executes the `Exec` or `NetExec` command to call an executable file, such as `Install.exe` or `Setup.exe`, that requires administrator permissions, the following events occur:

1. The following error code is output as a result of the `Exec` or `NetExec` command:
`0740: The requested action requires elevated privileges.`
2. The User Account Control dialog box for elevation of privileges appears.

Normally, the error in 1. occurs first after an `Exec` or `NetExec` command, and then the OS program compatibility assistant function is activated and the dialog box in 2. opens.

To prevent the error code and dialog box from appearing, you need to execute the script file as a user who has administrator permissions. For details about how to execute script files as a user who has administrator permissions, see [1.8.6 Execution as a user who has administrator permissions](#).

If you click the **Cancel** button in the User Account Control dialog box for elevation of privileges, the following error code might appear:

```
1223: The operation was canceled by the user.
```

(4) TerminateProcess command

The `TerminateProcess` command cannot forcibly terminate the processes being executed by another user (except for script processes). Even if you specify another user's process, the command will terminate normally without causing an error. This is the same operation as when a process specified by the process ID does not exist.

(5) GetProcessInfo command

The `GetProcessInfo` command can only acquire information about the processes being executed by the local user. If you specify the process ID of a process being executed by another user, an Access Denied error occurs. An Access Denied error also occurs if the process ID of a process being executed in the logon space is specified from a script that is being executed in the service space.

1.8.3 How to set up Manager

To set up information on the tabs in the dialog box that is opened by selecting **Tools** and then **Options** in the Script Manager window, you must run Manager as a user who has administrator permissions. For details about how to run Manager as a user who has administrator permissions, see *1.8.6 Execution as a user who has administrator permissions*.

1.8.4 How to change a registry entry

To set or change the value of a registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1/Script`, you need to run Registry Editor as a user who has administrator permissions. To do so, log on as a member of the Administrators group, and then start Registry Editor. After Registry Editor starts, the User Account Control dialog box for elevation of privileges will open. Click the **Continue** button.

1.8.5 Message box layout

The buttons in the message boxes displayed in JP1/Script by the `MessageBox` command are aligned to the right.

1.8.6 Execution as a user who has administrator permissions

If User Account Control (UAC) is disabled in Windows, you can perform operations with permissions of the logon user, but OS security will be degraded. Therefore, perform the following operations as much as possible.

(1) Starting Manager

After starting Manager as a user who has administrator permissions, execute script files and Editor functions as a user who has administrator permissions. In this case, you must also monitor script files from Editor as a user who has administrator permissions.

To start Manager as a user who has administrator permissions, use either of the following methods:

(a) Specifying "Run as administrator"

To specify the setting:

1. Right-click the shortcut of Manager to open the menu, and then select **Run as administrator**.
The User Account Control dialog box for elevation of privileges opens.
2. Click the **Continue** button.
3. If you are not a member of the Administrators group, enter the administrator password.

(b) Specifying "Run this program as an administrator" for the privilege level

To specify the setting:

1. Right-click the shortcut of Manager to open the menu, and then select **Properties**.
The Properties dialog box opens.
2. On the **Compatibility** page, select the **Run this program as an administrator** check box for the privilege level.

3. Start Manager.

The User Account Control dialog box for elevation of privileges opens.

If you are not a member of the Administrators group, the specification in 2. does not take effect. Instead, Manager is started with standard user privileges.

(2) Executing script files

To execute script files as a user who has administrator permissions, use either of the following methods.

(a) Specifying "Run this program as an administrator" for the privilege level

To specify the setting:

1. Right-click `sptxe.exe` in the JP1/Script installation folder to open the menu, and then select **Properties**.

The Properties dialog box opens.

2. On the **Compatibility** page, select the **Run this program as an administrator** check box for the privilege level.

3. Execute all the script files.

The User Account Control dialog box for elevation of privileges opens.

If you are not a member of the Administrators group, the specification in 2. does not take effect. Instead, the script files are executed with standard user privileges.

(b) Specifying "Run as administrator"

To specify the setting:

1. Create a shortcut of `sptxe.exe`, which is in the JP1/Script installation folder, on the desktop.

2. Right-click the shortcut to open the menu, and then select **Properties**.

The Properties dialog box opens.

3. On the **Shortcut** page, specify the script file name after `sptxe.exe` under **Target**.

4. Right-click the shortcut to open the menu, and then select **Run as administrator**.

`sptxe.exe` is executed, and the User Account Control dialog box for elevation of privileges opens.

5. Click the **Continue** button.

1.9 About the lock error retry and trial-open functions

The lock error retry function retries to access files and folders when a lock error[#] occurs during execution of a command or statement that accesses files and folders.

The trial-open function tries to open the file immediately after the `TextClose` command is executed to close a file to which data was written by the `TextOpen` command. This confirms that the data has been written to the disk.

Because typical Windows file systems use a disk write cache, a disk write delay might occur at the device level. Therefore, if a disk write delay has occurred, an attempt to execute the command or statement that accesses files and folders might fail due to a lock error. The lock error retry function and the trial-open function allow you to recover errors caused by a disk write delay during execution of a command or statement.

Note: A lock error code has occurred if either of the following error codes is output:

- 0032: A process cannot access a file because another process is using the file.
- 0005: Access was denied. Recheck file attributes or security.

1.9.1 Lock error retry function

If you access a file or folder immediately after its update, a temporary lock error might occur due to a disk write delay. When the lock error retry function is enabled, if a lock error occurs in a file or folder, a command retries file access. As a result, you no longer need to specify the retry processing in a script.

The lock error retry function retries access to a file or folder according to the specified retry count and retry wait time until the lock is cleared or another error occurs. The following describes the operation of the lock error retry function:

If the lock was cleared during retry and the file or folder was accessed normally:

The retry is terminated and processing of the command or statement continues.

If an error other than a lock error occurred during retry:

The retry is terminated, and execution of the command or statement is terminated due to the error.

If the lock was not cleared by the retry:

Execution of the command or statement is terminated due to the lock error.

Use the `SetRetryMode` command to retry only when a specific command is called. Even if the lock retry function is disabled, the `SetRetryMode` command allows the error retry function to take effect only when a specific command is called. You can also set the retry count and retry wait time for each command. For details about the commands that can be used with the lock error retry function, see [1.9.1\(2\) Commands and statements that can be used with the lock error retry function](#). For details about the `SetRetryMode` command, see [9.9.2 SetRetryMode \(set the lock error retry function\)](#).

(1) Retry count and retry wait time

The retry count and retry wait time are set as registry values.

(a) Retry count (units: number of times)

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX

Value name

IOErrorRetryCount

Value datatype

REG_DWORD

Value

- Specify a value in the range from 0 to 100. The default is 0.
- If 0 is specified, no retry is performed.
- If no value is set or if a value outside the range from 0 to 100 is specified, a default of 0 is assumed.

When the setting takes effect

The setting takes effect the next time the script file is executed.

(b) Retry wait time (units: seconds)

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX

Value name

IOErrorRetryWaitTime

Value datatype

REG_DWORD

Value

- Specify a value in the range from 1 to 60. The default is 1.
- If no value is set or if a value outside the range from 1 to 60 is specified, a default of 1 is assumed.

When the setting takes effect

The setting takes effect the next time the script file is executed.

(c) Extent of the settings

The settings are effective on all scripts that are executed after the values are set in the registry. The values set in the registry are effective only on the computer on which they are set.

(d) Changing the retry count and retry wait time

Use the `SetRetryMode` command to change the retry count and retry wait time set in the registry. The changes are effective until the `ResetRetryMode` command is executed.

(2) Commands and statements that can be used with the lock error retry function

The following table lists the commands and statements that can be used with the lock error retry function.

Table 1–6: Commands and statements that can be used with the lock error retry function

Type	Category	Command or statement
Basic command	Variable manipulation	SetGV
		GetGV
		DeleteGV
	File and folder management	IniRead
		IniWrite
		TextFileReplace
		TextOpen
		MakeDir
		DeleteDir
		DeleteFile
		Rename
		TempDir
		TempFile
		SetFileAttribute or SetFileAttr
		GetFileAttribute or GetFileAttr
		SetFileTime
		GetFileTime
		GetFileSize
		GetVersionInfo or GetVerInfo
		SplitFile
		CatFiles
		SetStandardFile or SetStdFile
		SetPath
	Copy	
	Message output	Message
	Menu display	Menu
	Evaluations	IsEmptyDir
		IsExistDir
		IsExistFile
		IsWriteableDir
		IsFileAttribute or IsFileAttr
		IsNew
	External program calls	Exec
		NetExec

Type	Category	Command or statement
Basic command	External program calls	CallSpt
	Automatic startup	EntryStartUp
		CancelStartUp
Special command	Registry operations	RegRead
		RegWrite
		RegDelete
		RegDeleteKey
	Graphics display	BitmapShow
	Shortcuts	MakeGroup
		DeleteGroup
		DeleteShortcut
	Process monitoring	GetProcessCount
		TerminateProcess
Statement		For...End For

1.9.2 Trial-open function

If enabled, the trial-open function tries to open a closed file according to the specified trial-open retry count and trial-open retry interval. This is in order to confirm that the data has been written to the disk. The function tries to open the file immediately after the `TextClose` command is executed. The function does not perform anything in the following cases:

- A file that was opened by using the `TextOpen` command with `ReadOnly` specified for the `Mode` parameter is closed by using the `TextClose` command.
- An attempt to close the file failed, causing the `TextClose` command to end with error.

If a lock error occurred during an attempt to open the file, the function retries opening the file until the lock is cleared or another error occurs.

The following describes the operation of the trial-open function:

If the lock was cleared during an attempt to open the file:

The trial-open is terminated and the `TextClose` command terminates normally.

If an error other than a lock error occurred during an attempt to open the file:

The trial-open is terminated and the `TextClose` command terminates normally.

If the lock was not cleared after the attempts to open the file:

The `TextClose` command terminates normally.

If you want to trial-open the file only after a specific `TextClose` command was executed, use the `SetTrialOpenMode` command according to [9.9.4 SetTrialOpenMode \(set the trial-open function\)](#). Even if the trial-open function is disabled, the `SetTrialOpenMode` command allows the trial-open function to take effect only after

a specific `TextClose` command is executed. You can also set the trial-open retry count and trial-open retry interval for each execution of the `SetTrialOpenMode` command.

(1) Trial-open retry count and trial-open retry interval

Set the trial-open retry count and the trial-open retry interval in the registry.

(a) Trial-open retry count (units: number of times)

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
TrialOpenRetryCount
```

Value datatype

```
REG_DWORD
```

Value

- Specify a value in the range from 0 to 100. The default is 0.
- If 0 is set, an attempt to open the file will not be made.
- If no value is set or if a value outside the range from 0 to 100 is specified, a default of 0 is assumed.

When the setting takes effect

The setting takes effect the next time the script file is executed.

(b) Trial-open retry interval (units: milliseconds)

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
TrialOpenRetryWaitTime
```

Value datatype

```
REG_DWORD
```

Value

- Specify a value in the range from 100 to 60,000. The default is 100.
- If no value is set or if a value outside the range from 100 to 60,000 is specified, a default of 100 is assumed.

When the setting takes effect

The setting takes effect the next time the script file is executed.

(c) Scope of the settings

The settings are effective on all scripts that are executed after the values are set in the registry.

The values set in the registry are effective only on the computer on which they are set.

(d) Changing the trial-open retry count and trial-open retry interval

Use the `SetTrialOpenMode` command to change the trial-open retry count and trial-open retry interval set in the registry. The changes are effective until the `ResetTrialOpenMode` command is executed.

2

Preparation for Using JP1/Script

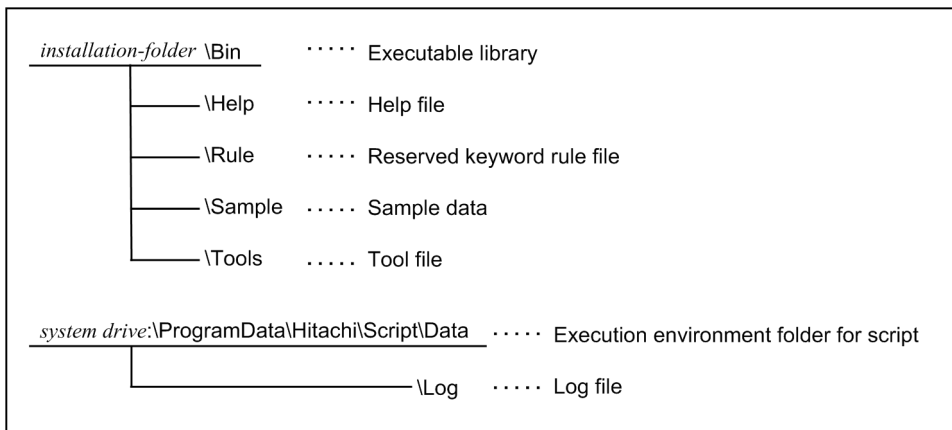
This chapter explains how to install and uninstall JP1/Script, the accounts used for communication with other computers, and the environment setup procedure in various system environments.

2.1 Installing and uninstalling JP1/Script

This section describes how to install JP1/Script, how to register the JP1/Script service, and how to uninstall JP1/Script.

2.1.1 Program installation folders

Install the JP1/Script programs in the following folders:

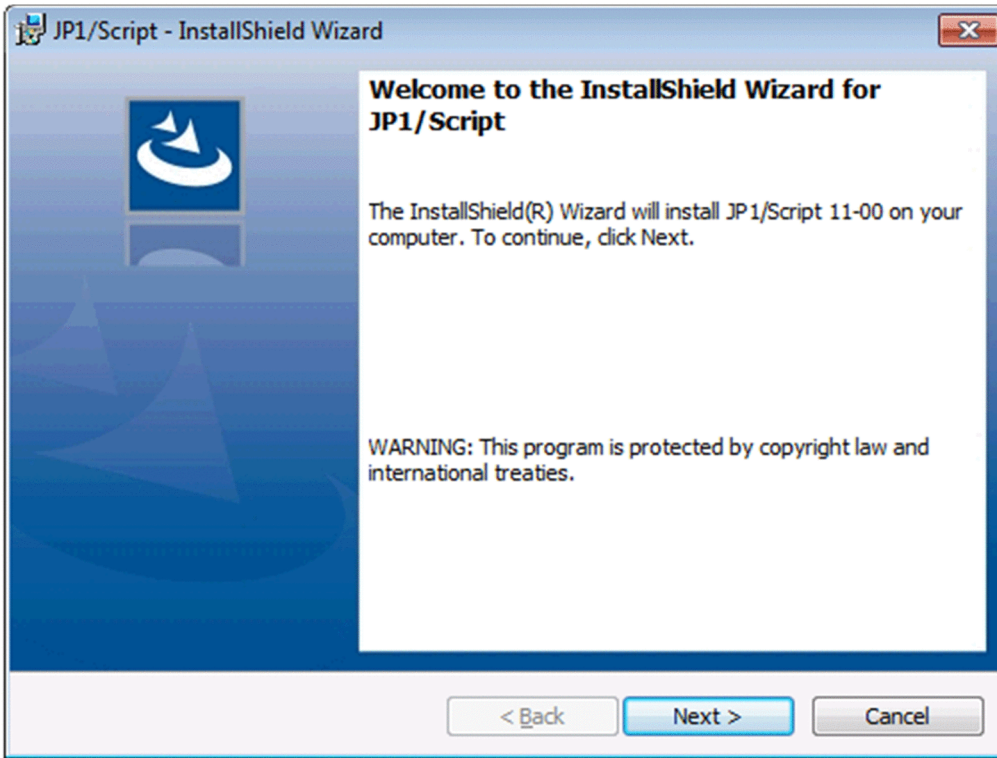


The execution environment folders for scripts (\DATA) and log data (\LOG) are unconditionally configured as shown above (you cannot change them during installation). If you want to change their locations (for example, because the system drive does not have enough free space), in the Manager window, select **Tools** and then **Options** to open the Options dialog box. In this dialog box, change the **Cluster Environment** settings. Note, however, that you cannot place the folders in the Program Files folder on the system drive. If you do so, the program will no longer operate correctly because the folders in the Program Files folder will be redirected by the OS.

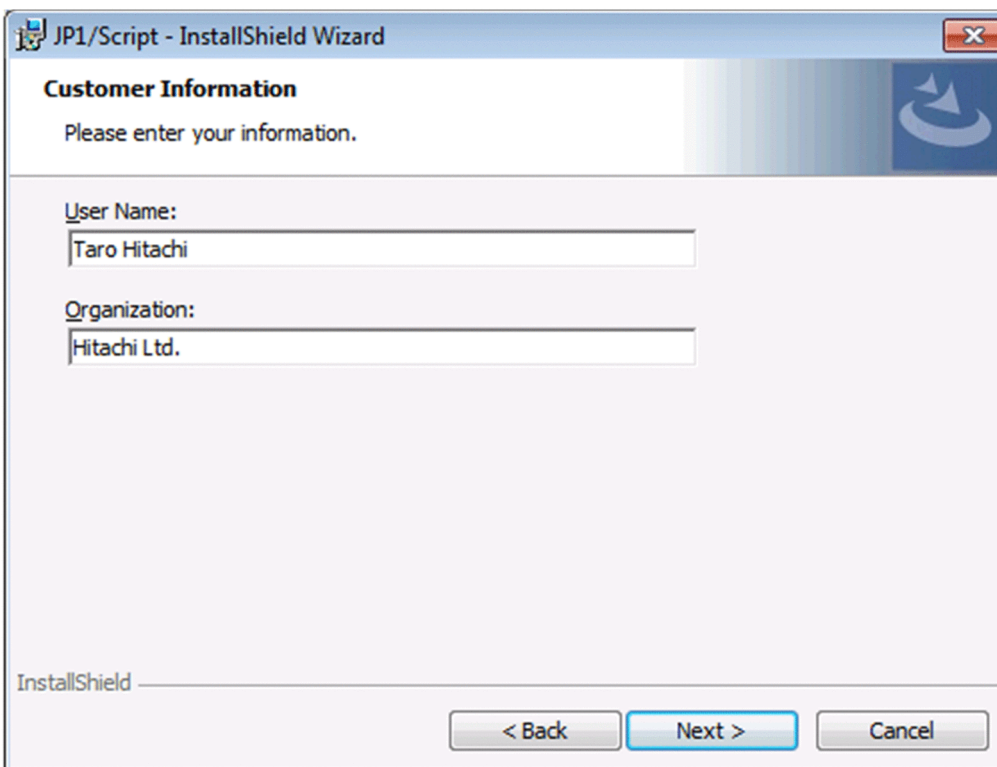
2.1.2 Installing JP1/Script

To install JP1/Script:

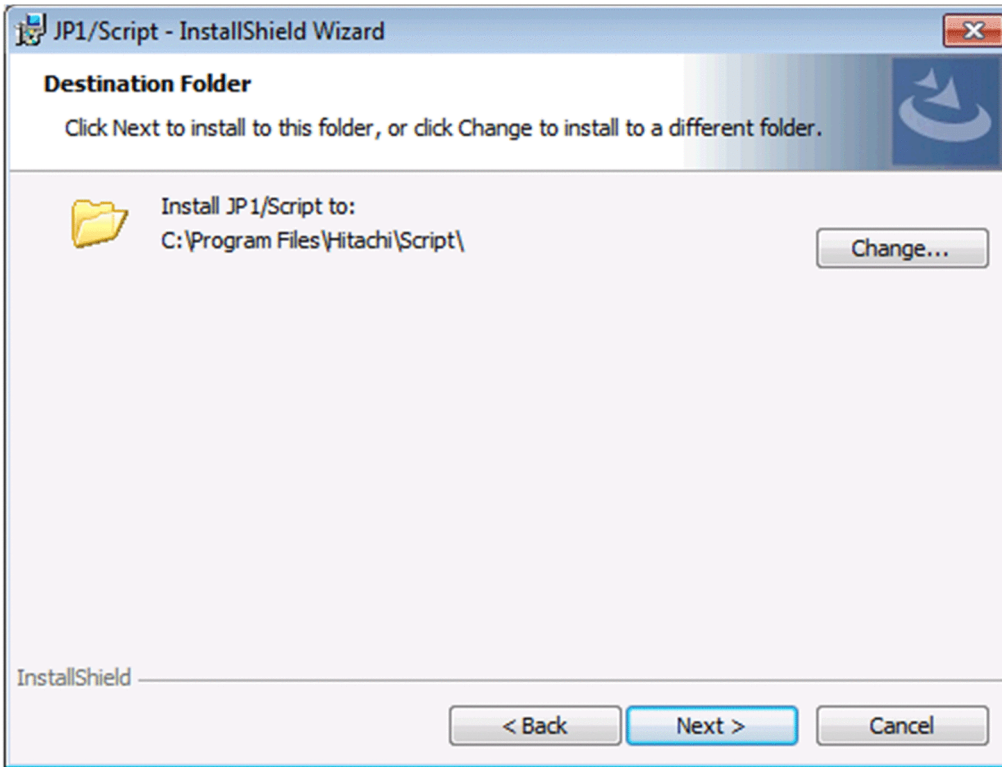
1. Before you install JP1/Script, log on with administrative privileges.
2. Set the JP1/Script installation medium.
The JP1/Script Install dialog box appears.
3. Check the program that you are about to install, then click the **Next** button.



4. Enter the user information (your user name and company name), then click **Next**. The names that you entered are set in a registry file as user information.

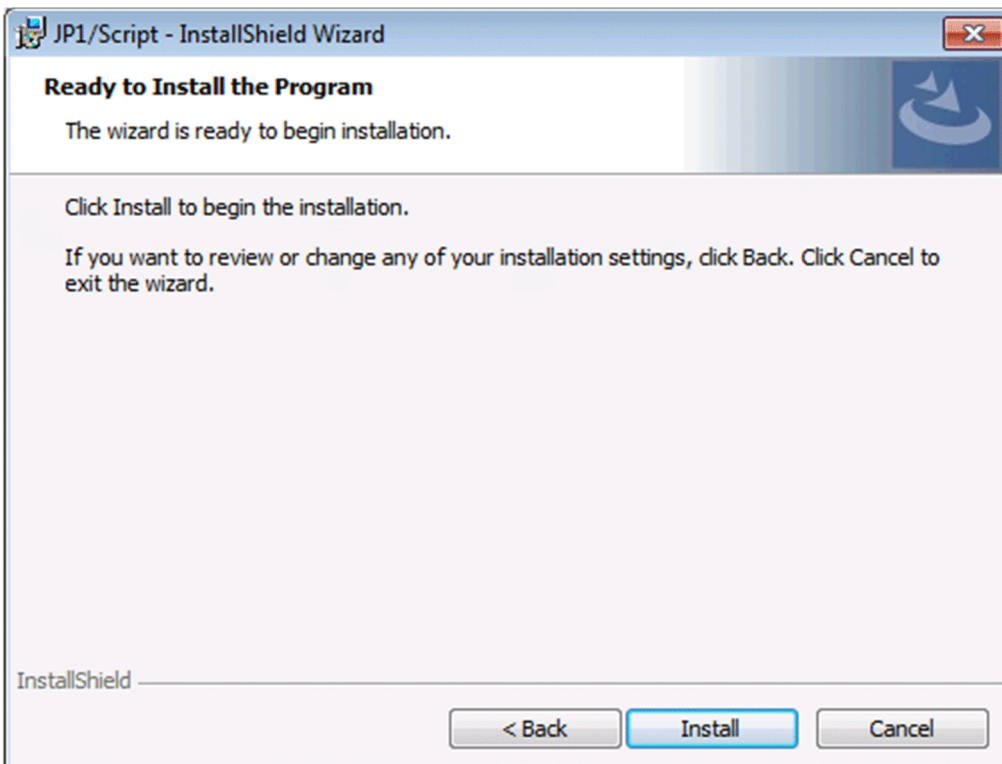


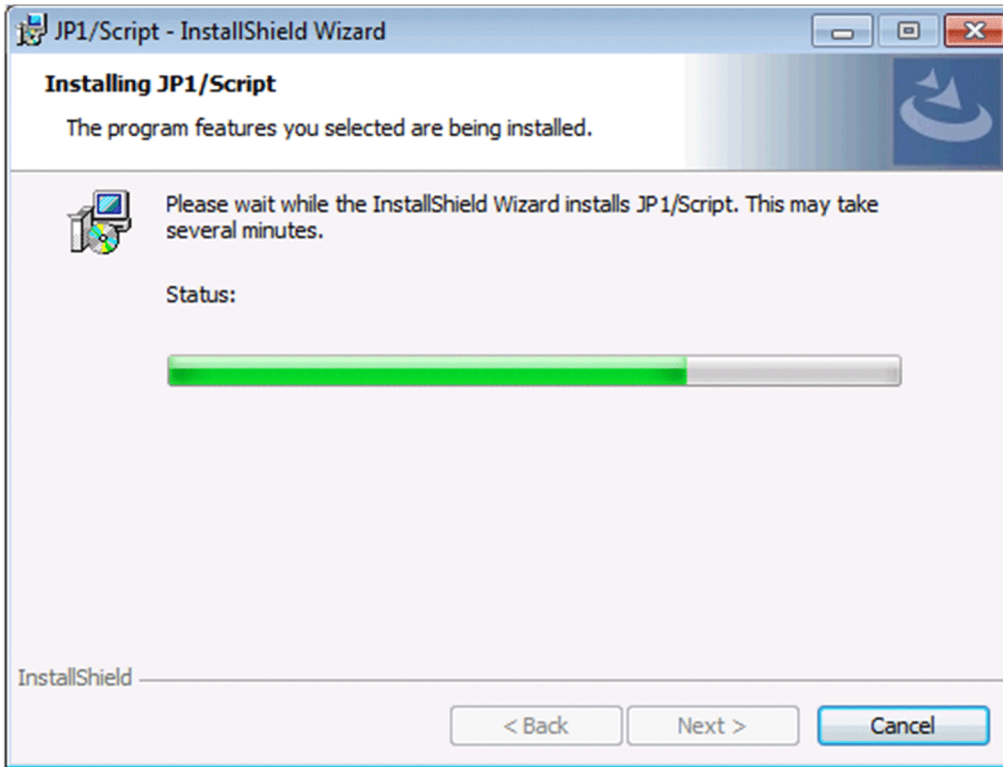
5. Specify a folder in which to install JP1/Script, then click **Next**.
To change the folder in which to install JP1/Script, click the **Change** button and then specify a folder. If you type a non-existent folder, a confirmation message appears and the installer automatically creates the specified folder.



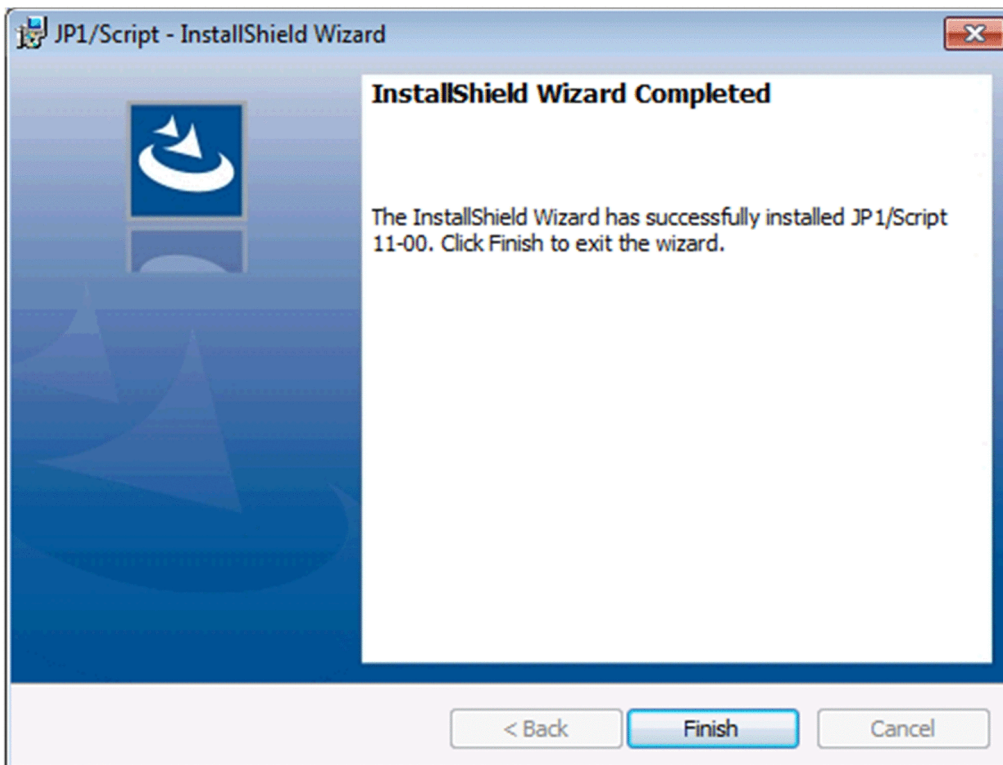
6. Click the **Install** button.

Installation starts, and a dialog box indicating installation progress appears.





7. In the Install Complete dialog box, click the **Finish** button.



8. Restart the computer.

Remote installation (software distribution) using JP1/Software Distribution

JP1/Script supports remote installation by JP1/Software Distribution.

For details about the actual procedure for remote installation when JP1/Software Distribution is used, see the *JP1/Software Distribution Administrator's Guide Volume 1 (For Windows Systems)*.

2.1.3 Registering the JP1/Script service

The JP1/Script service account must have the Log On as a Service right.

The system account is set when JP1/Script is installed. To change the account name, in Windows **Control Panel**, select **Administrative Tools**, and then double-click **Services**. In the Services window that opens, select the JP1/Script service and change the properties. To use the `SetGV`, `GetGV`, `DeleteGV`, or `NetExec` command to call a remote computer from the JP1/Script service, assign an appropriate account to the JP1/Script service that will allow you to log on to the computer you want to call. Unless an appropriate account is assigned, these commands will not function properly. For details about logon accounts, see [2.2 Accounts for communication with other computers](#).

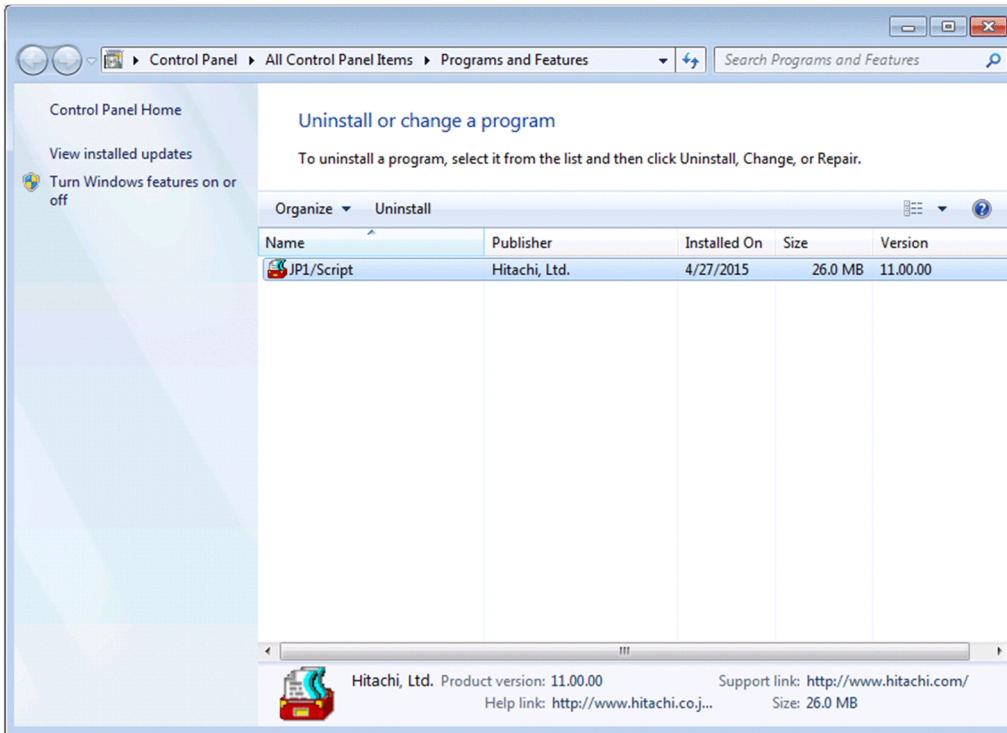
Important note

- If you execute a JP1/Script service with the system account, system resources might be shared with other service programs that are executed with the system account and operation might be affected. If any other service program is being executed with the system account, specify an appropriate account.
- When you perform a reinstallation or upgrade installation, the existing account name is inherited.

2.1.4 Uninstalling JP1/Script

To uninstall JP1/Script:

1. Log on with administrative privileges.
If you logged on using an account without administrative privileges, uninstallation will fail.
2. Exit all active JP1/Script programs.
Uninstallation will not succeed unless all programs are terminated.
3. In Windows **Control Panel**, select **Programs**, and then **Programs and Features**.
A list of programs appears.



4. From the list of programs, select **JP1/Script**, and then click **Uninstall**. Alternatively, right-click **JP1/Script** to open the menu, and then select **Uninstall**.

JP1/Script is uninstalled.

5. Restart the computer.

Important note

- All files and folders created by the JP1/Script installer are deleted.
- If another application is accessing a file in the JP1/Script installation folder, the installation folder might not be deleted when JP1/Script is uninstalled.
- The files, folders, and registry keys and entries listed below are not deleted by uninstalling JP1Script. Delete any unnecessary files, folders, or registry keys or entries after uninstallation.
 - Files and folders created in *system-drive*\ProgramData\Hitachi\Script after installation
 - Keys and entries created under HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1/Script (for a 64-bit version of Windows, in HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\HITACHI\JP1/Script registry)
 - Keys and entries created under the HKEY_CURRENT_USER\Software\JP1/Script registry

2.1.5 Reinstalling JP1/Script

Reinstallation installs the same version of JP1/Script as the one that is already installed. The existing JP1/Script installation folder and settings will be inherited after reinstallation.

The following describes how to reinstall JP1/Script.

To reinstall JP1/Script:

1. If the output folder for JP1/Script management files has been changed to a folder on the shared disk, make the shared disk online.
If you install JP1/Script with the shared disk offline, automatic startup setting might be disabled.
2. Log on with administrator permissions.
If you logged on using an account without administrator permissions, installation will fail.
3. Exit all active JP1/Script programs.
Installation will not succeed unless all programs are terminated.
4. Set the JP1/Script installation medium.
The JP1/Script Install dialog box appears.
5. Check the program that you want to install, and then click the **Next** button.
6. Click the **Install** button.
Installation starts, and the dialog box indicating the progress of installation appears.
7. In the Install Complete dialog box, click the **Finish** button.
8. Restart the computer.

Important note

Reinstalling JP1/Script deletes the correction patches that have been applied. Apply the patches again after reinstallation.

Remote installation (software distribution) using JP1/Software Distribution

JP1/Script supports remote installation by JP1/Software Distribution.

For details about the actual procedure for remote installation when JP1/Software Distribution is used, see the *JP1/Software Distribution Administrator's Guide Volume 1 (For Windows Systems)*.

2.1.6 Upgrade installation of JP1/Script

Upgrade installation installs a later version of JP1/Script than the one that is already installed. The existing JP1/Script installation folder and settings will be inherited after upgrade installation.

Important note

- Before upgrading a version earlier than 10-00, back up the operating environment information and management files. An upgrade installation first uninstalls the existing version of JP1/Script. If an error occurs during uninstallation, the JP1/Script operating environment information and management files might not be installed again after they are deleted. If an error should occur during uninstallation, reinstall the previous version of JP1/Script, and then perform an upgrade installation again.
- You cannot perform an upgrade installation for a version earlier than 08-00. You will have to uninstall the existing JP1/Script, and then install the new version of JP1/Script.

The following describes how to perform an upgrade installation of JP1/Script.

To perform an upgrade installation of JP1/Script:

1. If the output folder for JP1/Script management files has been changed to a folder on the shared disk, make the shared disk online.
If you install JP1/Script with the shared disk offline, automatic startup setting might be disabled.
2. Log on with administrator permissions.
If you logged on using an account without administrator permissions, installation will fail.
3. Exit all active JP1/Script programs.
Installation will not succeed unless all programs are terminated.
4. Set the JP1/Script installation medium.
The JP1/Script Install dialog box appears.
5. Check the program that you want to upgrade, and then click the **Next** button.
6. Click the **Install** button.
Installation starts, and the dialog box indicating the progress of installation appears.
7. In the Install Complete dialog box, click the **Finish** button.
8. Restart the computer.

Remote installation (software distribution) using JP1/Software Distribution

JP1/Script supports remote installation by JP1/Software Distribution.

For details about the actual procedure for remote installation when JP1/Software Distribution is used, see the *JP1/Software Distribution Administrator's Guide Volume 1 (For Windows Systems)*.

2.1.7 Downgrade installation

You cannot perform a downgrade installation of JP1/Script to a version earlier than the one currently installed. To downgrade JP1/Script, uninstall the existing JP1/Script, and then install the older version.

2.2 Accounts for communication with other computers

This section describes the accounts used for communication with other computers.

To use the `SetGV`, `GetGV`, `DeleteGV`, or `NetExec` command to call a remote computer, you must use an appropriate account and password to execute the script that calls the commands. Unless you use an appropriate account and password that allows you to log on to the called computer, the commands will not function properly.

Table 2-1 lists the accounts and passwords that are used for executing scripts that call commands.

Table 2–1: Accounts and passwords used for executing scripts that call commands

No.	Script execution method	Account and password used for executing the caller script
1	Script is executed from the JP1/Script service by the automatic startup function.	Logon account and password that were set for the JP1/Script service.#
2	Script is executed from a JP1/AJS service program.	OS user's account and password that were mapped to the JP1 user executing the script.
3	Script is executed from a service program other than JP1/AJS.	Logon account and password that were set for the service.#
4	Script is executed from Script Launcher by the automatic startup function.	Account and password that were used to log on to the computer.
5	Script is executed from Script Manager or Script Editor.	
6	Script is executed directly from Windows Explorer or another program.	

#

You cannot use the system account as the service account when using the `SetGV`, `GetGV`, `DeleteGV`, or `NetExec` command to call a remote computer. A user account that exists on the called computer must be set for the service account.

Table 2-2 lists the execution accounts and passwords that are used when the connection is successful and the `NetExec` command executes an executable file on the called computer.

The execution accounts and passwords in Table 2-2 are also used when, for example, the `NetExec` command calls a copy operation (including the `Copy` command) to the file server on the domain.

Table 2–2: Accounts and passwords of executable files called from the NetExec command

No.	Value of the fifth argument of the NetExec command	Account and password used for the called executable files
1	True (execute in service space)	Logon account and password of Script service in the called computer
2	False (execute in logon space)	Account and password used to log on to the called computer

For details about the `SetGV`, `GetGV`, and `DeleteGV` commands, see [8.2.5 SetGV \(set a global variable\)](#), [8.2.6 GetGV \(get a global variable\)](#), and [8.2.7 DeleteGV \(delete a global variable\)](#).

For details about the `NetExec` command, see [8.10.2 NetExec \(call an executable file on the local PC or remote PC\)](#).

2.3 Environment setup in a cluster system environment

In a cluster system environment, the active system can execute jobs while the standby system is in standby status until failover occurs. You can use JP1/Script in such an environment by means of an active-standby configuration, but the script being executed at the time of failover cannot be inherited.

However, the execution environment at a time of failover can be inherited. This section describes how to set up an environment to allow the execution environment to be inherited when a failover occurs.

2.3.1 Setting up an environment for using a Script Launcher

The following describes how to set up an environment for using a Script Launcher.

Perform steps 3 through 5 on either the active system or standby system in a cluster system environment. Other steps must be performed on both systems.

To use the `NetExec` command to execute an executable file in the logon space, you must be logged on to Windows in both the active and standby systems, and the Script Launcher must be running.

To set up an operating environment:

1. Install JP1/Script.

Install JP1/Script. Specify a folder on the local drive as the installation folder.

2. Change the JP1/Script service startup type to manual.

In Windows **Control Panel**, select **Administrative Tools**, and then double-click **Services**. In the Services window that opens, select the JP1/Script service and display its properties. Then change the startup type from **Auto** to **Manual**.

3. Decide which shared disk to use.

Decide which shared disk to specify at step 6 for output of the JP1/Script management files.

4. Start Cluster Administrator.

From the Windows **Start** menu, choose **Programs, Administrative Tools**. Then double-click **Cluster Administrator** to open the Cluster Administrator window.

5. Create resources.

Create the following three resources in the Cluster Administrator window:

(1) Shared disk resource determined at step 3.

- Resource type: Physical Disk
- Resource name: Any

(2) Logical IP address resource

- Resource type: IP Address
- Resource name: Any
- Resource dependency: Resource of (1)

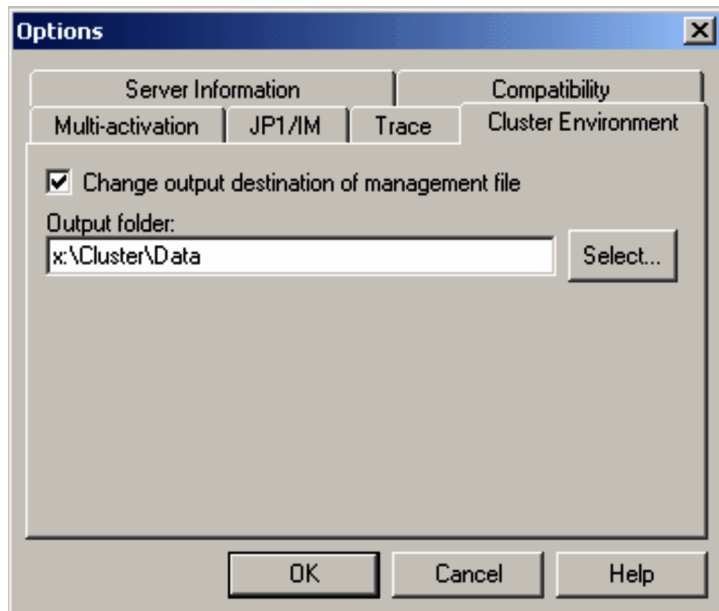
(3) JP1/Script service resource

- Resource type: Generic Service

- Resource name: Any
- Resource dependencies: Resources (1) and (2)
- Service name: JP1_Script

6. Change the output folder for JP1/Script management files to a folder on the shared disk.

In the Manager window, choose **Tools** and then **Options** to open the Options (Clustering Environment) dialog box. In this dialog box, change the file output destination to the folder on the shared disk determined at step 3.



7. Restart the system.

Restart the system to enable the settings you entered at step 6.

Note

If an active-active configuration is used, that is, both systems in the cluster system environment are active, you can use JP1/Script, but the script being executed at the time of failover cannot be inherited. In such an environment, do not set up the environment described above because the execution environment at the time of failover can be inherited.

2.3.2 Setting up an environment for using a Script Launcher service

The following describes how to set up an environment for using a Script Launcher service.

Perform steps 4 through 6 on either the active system or standby system in a cluster system environment. Other steps must be performed on both systems. To use the Script Launcher service in a cluster system environment, you must have logged on to both the active and standby systems by using the actual host name.

To set up an operating environment:

1. Install JP1/Script.
Install JP1/Script. Specify a folder on the local drive as the installation folder.
2. Change the JP1/Script service startup type to manual.

In Windows **Control Panel**, select **Administrative Tools**, and then double-click **Services**. In the Services window that opens, select the JP1/Script service and display its properties. Then change the startup type from **Auto** to **Manual**.

3. Change the Script Launcher service startup type to manual.

In Windows **Control Panel**, select **Administrative Tools**, and then double-click **Services**. In the Services window that opens, select the Script Launcher service and display its properties. Then change the startup type from **Auto** to **Manual**.

4. Decide which shared disk to use.

Decide which shared disk to specify at step 7. for output of the JP1/Script management files.

5. Start Cluster Administrator.

From the Windows **Start** menu, choose **Programs, Administrative Tools**. Then double-click **Cluster Administrator** to open the Cluster Administrator window.

6. Create resources.

Create the following four resources in the Cluster Administrator window:

(1) Shared disk resource determined at step 4.

- Resource type: Physical Disk
- Resource name: Any

(2) Logical IP address resource

- Resource type: IP Address
- Resource name: Any
- Resource dependency: Resource of (1)

(3) JP1/Script service resource

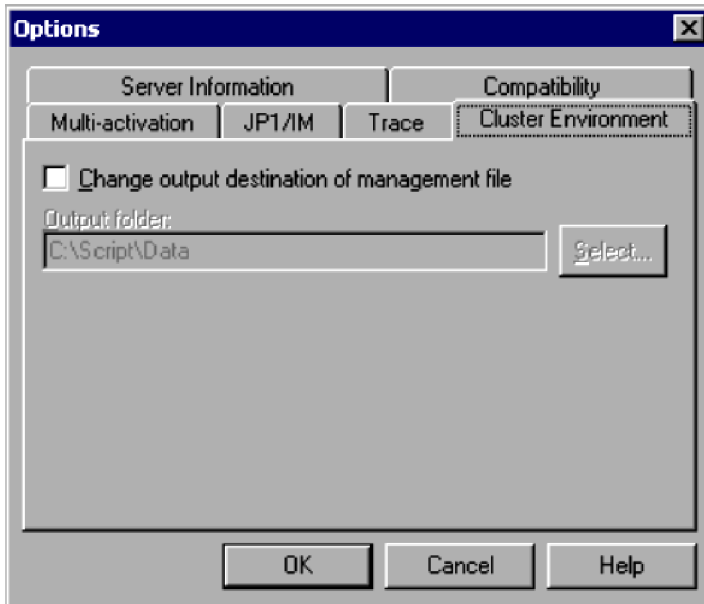
- Resource type: Generic Service
- Resource name: Any
- Resource dependencies: Resources (1) and (2)
- Service name: JP1_Script

(4) Script Launcher service resource

- Resource type: Generic Service
- Resource name: Any
- Resource dependencies: Resources (1), (2), and (3)
- Service name: JP1_SCRIPT_LAUNCHER
- Start parameters: Logon user name

7. Change the output folder for JP1/Script management files to a folder on the shared disk.

In the Manager window, choose **Tools** and then **Options** to open the Options (Clustering Environment) dialog box. In this dialog box, change the file output destination to the folder on the shared disk determined at step 4.



8. Restart the system.

Restart the system to enable the settings you entered at step 7.

Note

If an active-active configuration is used, that is, both systems in the cluster system environment are active, you can use JP1/Script, but the script being executed at the time of failover cannot be inherited. In such an environment, do not set up the environment described above because the execution environment at the time of failover can be inherited.

2.4 Environment setup in a Remote Desktop service environment

You can use JP1/Script on the Windows Server 2008 R2 Remote Desktop service, Windows Server 2012 Remote Desktop service, or Windows Server 2012 R2 Remote Desktop service. To allow users other than an administrator to log in through the remote session and use JP1/Script, specify the Windows security settings to assign the **Create global objects** permission to the login users. Note the following when using JP1/Script in a Remote Desktop service environment:

- Script Launcher can be started in the primary session on the Remote Desktop Session Host server, and cannot be started in the remote session on the client. The JP1/Script service can be started and stopped in both sessions.
- Script files registered for automatic startup are started in the primary session on the Remote Desktop Session Host server irrespective of the startup type.
- If you want to use the remote session on the client to start the executable file that will be called by the `NetExec` command, use the Script Launcher service. For details about the Script Launcher service, see [1.7.6 About the Script Launcher service](#).

Important note

Do not use the `Beep` command in the remote session on the client. If you do so, an error occurs. For example, `0005 (Access was denied. Recheck file attributes or security)` is output.

Note

Use `RemoteApp` to specify which of the JP1/Script programs installed on the Remote Desktop Session Host server will be accessible from users.

Because programs started via `RemoteApp` are run on a remote server, reserved variables are the same as those on the remote server. Operation commands also run on the remote server. Because the programs started on the remote server and those started on the client are displayed in the same way on the desktop, be careful not to confuse the programs.

2.5 Renaming hosts and changing IP addresses

To rename a host or change its IP address:

1. Make sure that no PC is running a script on the PC whose host is to be renamed or IP address is to be changed.
2. Stop the JP1/Script service and Script launcher or Script launcher service.
To stop the Script launcher, add the command line `/INST` to `Spthstp.exe` in the *JP1/Script-installation-folder* \Bin folder, and then execute the program.
3. After making the change, start the JP1/Script service and Script launcher or Script launcher service.
To start Script Launcher, from the Windows **Start** menu, choose **Programs, JP1/Script**, and then **Script Launcher**.

Notes

- If a computer name has been specified in the `NetExec`, `GetGV`, `SetGV`, or `DeleteGV` command in the format of a fully qualified domain name (FQDN), you must change the specified computer name when the domain name is changed.
- If you have set a user account in the JP1/Script service account, check and, if necessary, revise the account if you change the domain name.

2.6 Monitoring of JP1/Script by the activity monitoring program

To use the process monitoring function to monitor JP1/Script processes, set JP1/PFM/SSO process monitoring conditions according to the following process monitoring information:

- Name of the application to be monitored: JP1/Script V11.0 (Windows)
- Additional information: JP1/Script (Windows)
- Number of processes to be monitored: 2

Parent process					Child process				
Number of processes	Type#	Process name	Threshold values		Number of processes	Type# ¹	Process name	Threshold values	
			Min.	Max.				Min.	Max.
3	2	SPTHSV	1	1	1	2	SPTTMS	1	1
	2	Spthlnch	1 ^{#2}	1 ^{#3}	0	--	--	--	--
	2	SPTHLHSV	1	1	0	--	--	--	--

Legend:

--: Not applicable

#1:

Type of process name (1: command line name; 2: executable file name)

#2:

Because the Spthlnch process is automatically started from startup at logon and automatically terminates at logoff, the minimum threshold at logoff is 0.

#3:

To use the Remote Desktop service to log on, specify 1 plus the number of users who can concurrently log on by using the Remote Desktop service. To enable Windows Fast User Switching, specify the maximum number of users who can concurrently log on. To use the Remote Desktop service and enable Windows Fast User Switching, specify the sum total of the following: 1 + number of users who can concurrently log on by using the Remote Desktop service + maximum number of users who can concurrently log on.

For details about the setup method, see the manual *JP1/Performance Management/SNMP System Observer Description, Operator's Guide and Reference*.

JP1/PFM/SSO is the successor of JP1/SSO. If you are using JP1/SSO, see the manual *JP1/Server System Observer Description, User's Guide and Reference*.

One of the following conditions for monitoring the SPTHSV process should be applicable:

- **Service** is set as the **Start type**, and the script is started automatically.
- A command for manipulating variables (SetGV, GetGV, DeleteGV) is executed from another computer.
- Process Viewer or Trace Viewer is used from another computer.
- The NetExec command with a process-monitoring computer specified for the first argument and True specified for the fifth argument has been executed on the local computer or remote computer.
- The Message command with Target_File, Target_SPAFile, or Target_SPXFile specified is executed.
- The EntryStartUp or CancelStartUp command is executed.
- A command for manipulating variables is executed.

If none of these conditions is satisfied, there is no need to monitor the process.

One of the following conditions for monitoring the `Spthlnch` process should be applicable:

- **Logon** is set as the **Start type**, and the script is started automatically.
- The `NetExec` command with a process-monitoring computer specified for the first argument and `False` specified for the fifth argument has been executed on the local computer or remote computer.

If none of these conditions is satisfied, there is no need to monitor the process.

The following condition for monitoring the `SPTHLSV` process should be applicable:

- The `NetExec` command with a process-monitoring computer specified for the first argument and `False` specified for the fifth argument has been executed on the local computer or remote computer.

If none of these conditions is satisfied, there is no need to monitor the process.

2.7 Starting and stopping JP1/Script

This section describes how to start and stop JP1/Script.

2.7.1 Starting JP1/Script

Start JP1/Script component programs as described below.

(1) Starting Manager

To create or edit a script file using JP1/Script, first start Manager:

1. From the Windows **Start** menu, choose **Programs, JP1_Script**, then **Manager**

Manager functions include script file management, setting of the execution environment, and checking script syntax.

For details about Manager, see *3.1.1 Script Manager window and menus*.

(2) Starting Editor

There are three ways of starting Editor.

(a) Starting from the Windows Start menu

1. From the Windows **Start** menu, choose **Programs, JP1_Script**, then **Editor**.

(b) Starting from Manager

Start Editor to actually create or edit a script file:

1. Select a script file in the Manager window.
2. In the Manager window, choose **File, Edit**.
If a different editor is associated with the selected file, that editor will open.

(c) Drag-and-drop operation

1. Drag a script file icon from Windows Explorer, and drop it on the Editor icon or in the active Editor window.
For details about Editor, see *3.2.1 Script Editor window and menus*.

(3) Starting Trace Viewer

The traces produced at script file execution are output to a trace file. To view trace status, start Trace Viewer in either of the following ways.

(a) Starting from the Windows Start menu

1. From the Windows **Start** menu, choose **Programs, JP1_Script**, then **Trace Viewer**.

(b) Starting from Manager

1. In the Manager window, choose **File, Start Trace Viewer**.

Trace Viewer starts and the Trace Viewer window opens, displaying a list of trace files.

For details about Trace Viewer, see [3.4.1 Script Trace Viewer window and menus](#).

(4) Starting Menu Editor

There are five ways of starting Menu Editor.

(a) Starting from the Windows Start menu

1. From the Windows **Start** menu, choose **Programs, JP1_Script**, then **Menu Editor**.

(b) Starting from Manager

1. Select a script file in the Manager window.
2. From the **Tools** menu, choose **Start Menu Editor**. Alternatively, right-click to open the pop-up menu, then choose **Start Menu Editor**.

(c) Starting from Editor

1. From the **Edit** menu, choose **Menu Editor**. Alternatively, right-click to open the pop-up menu, then choose **Menu Editor**.

(d) Drag-and-drop operation

1. Drag a menu information file from Windows Explorer, and drop it on the Menu Editor icon or in the active Menu Editor window.
If the menu information file is being updated, when you drop the file a message asks if you want to save the file.

(e) Starting from Windows Explorer

1. In Windows Explorer, select a menu information file.
2. Choose **File, Open**. Alternatively, double-click the selected file.

For details about Menu Editor, see [3.6.1 Script Menu Editor window and menus](#).

(5) Automatic startup

You can use the JP1/Script service or Script Launcher to automatically start a script to use JP1/Script. Automatic startup is not applicable to the Script Launcher service.

There are two ways of starting a script file automatically:

- By starting in the service space
- By starting in the logon space

To enable automatic startup, you must set the execution environment (startup information) in Manager. For details, see [3.1.14 Setting a script to start automatically](#).

2.7.2 Stopping JP1/Script

Stop the JP1/Script component programs as described below.

(1) Stopping Manager

1. In the Script Manager window, choose **File, Exit**.
Manager stops.

(2) Stopping Editor

1. In the Script Editor window, choose **File, Exit**.
Editor stops.

(3) Stopping Trace Viewer

1. In the Script Trace Viewer window, choose **File, Exit**.
Trace Viewer stops.

(4) Stopping Menu Editor

1. In the Script Menu Editor window, choose **File, Exit**.
Menu Editor stops.

(5) Terminating the script file forcibly

You can use Process Viewer, the `TerminateProcess` command, the `WaitForExec` command (with `Abort` specified), or the `Exit` command (with `Abort` specified) to forcibly terminate a script file. If the script file that is to be terminated forcibly has called another script file, the called script file will also be terminated forcibly. If an executable file other than a script file is called, only that executable file is terminated forcibly.

(6) Forcibly terminating JP1/Script from JP1/AJS

JP1/Script script files registered in a JP1/AJS jobnet or job can be forcibly terminated from JP1/AJS. At this time, executable files that were called from the JP1/Script script files are also terminated.

The range of executable files that are terminated depends on whether they were called from a JP1/Script script file or from a non-JP1/Script script file. The following table illustrates three specific patterns:

Pattern	Calling procedure	Range of executable files terminated when JP1/Script script file is forcibly terminated from JP1/AJS
Pattern 1	1: JP1/AJS process → 2: JP1/Script script file → 3: Non-JP1/Script script file	Up to executable file 3.
Pattern 2	1: JP1/AJS process → 2: JP1/Script script file → 3: JP1/Script script file → 4: Non-JP1/Script script file	Up to executable file 4.
Pattern 3	1: JP1/AJS process → 2: JP1/Script script file → 3: Non-JP1/Script script file → 4: Non-JP1/Script script file	Up to executable file 3. (Executable file 4 is not terminated.)

However, if the JP1/Script script file does not terminate within 30 seconds after forced termination is requested from JP1/AJS, use the `TerminateProcess` function of the Win32 API to forcibly terminate it. Because the `TerminateProcess` function does not terminate the executable file called from a JP1/Script script file, only the JP1/Script script file of 2 in each pattern shown in the table above is terminated.

Important note

Forced termination should be used only when developing a script. Forced termination is not recommended for job execution.

2.8 Changing the system date and time

Perform the procedure below to change the system date and time.

To change the system date and time:

1. Stop the JP1/Script service and Script Launcher.

To stop Script Launcher, add the command line `/INST` to `Spthst.exe`, which is in the JP1/Script installation folder, and then execute the file.

2. Stop all running script files.

Use JP1/Script Process Viewer to make sure that all script files are stopped. If necessary, forcibly terminate script files from JP1/Script Process Viewer.

3. Change the system date and time, and then start the JP1/Script service and Script Launcher.

To start Script Launcher, from the Windows **Start** menu, choose **Programs, JP1/Script**, and then **Script Launcher**.

2.9 Setting users permitted to remotely execute the NetExec command

To set the users permitted to remotely execute the `NetExec` command:

1. Select the users you want to permit to remotely execute the `NetExec` command.
Select the users for each space (service space or logon space) in which executable files called by the `NetExec` command are executed.
2. Create a folder in which the `NetExec` command restriction policy files will be stored.
Create a `Conf` folder in the `JP1/Script` installation folder.
3. Create the `NetExec` command restriction policy files.
Create the following files for each space (service space and logon space) in which executable files called by the `NetExec` command are executed.
 - For the service space, create the `SPTHSV_ACP` file.
 - For the logon space, create the `SPTHLSV_ACP` file.
4. Place the created `NetExec` command restriction policy files in the `JP1/Script-installation-folder\Conf` folder.
Place the `SPTHSV_ACP` file and `SPTHLSV_ACP` file you created in step 3 in the `JP1/Script-installation-folder\Conf` folder.
5. In the `NetExec` command restriction policy file, set the users permitted to remotely execute the `NetExec` command.
Perform the following procedure:
 1. Right-click the `NetExec` command restriction policy file to open the menu, and then select **Properties**.
The Properties dialog box opens.
 2. On the **Security** page, click the **Add** button, and then select users you want to permit to remotely execute the `NetExec` command.
In the Select Users or Groups dialog box that opens, select the users you want to permit to remotely execute the `NetExec` command.
 3. On the **Security** page, select the users you added, and then specify the following information for **Permissions**:
 - Read: Permitted
 - Write: Permitted
 4. Make sure that unnecessary users are not set.
Delete any unnecessary users.
6. In the `NetExec` command restriction policy file, set the file management user.
Perform the following procedure:
 1. Right-click the `NetExec` command restriction policy file to open the menu, and then select **Properties**.
The Properties dialog box opens.
 2. On the **Security** page, click the **Add** button, and then select the file management user.
In the Select Users or Groups dialog box that opens, select the file management user.
 3. On the **Security** page, select the user you added, and then specify the following information for **Permissions**:
Full control: Permitted

7. In the `NetExec` command restriction policy file, set the logon accounts for the JP1/Script service and Script Launcher service.

To set the user who starts the JP1/Script service, perform the procedure below for the `SPTHSV_ACP` file. To set user who starts the Script Launcher service, perform the following procedure for the `SPTHLSV_ACP` file.

1. Right-click the `NetExec` command restriction policy file to open the menu, and then select **Properties**.

The Properties dialog box opens.

2. On the **Security** page, click the **Add** button, and then select the logon account for either the JP1/Script service or the Script Launcher service.

In the Select Users or Groups dialog box that opens, select the user who starts the service.

3. On the **Security** page, select the user you added, and then specify the following information for **Permissions**:

- Read: Permitted
- Write: Permitted

8. In the folder that contains the `NetExec` command restriction policy file, set the logon accounts for the JP1/Script service and the Script Launcher service.

Perform the following procedure:

1. Right-click the `Conf` folder to open the menu, and then select **Properties**.

The Properties dialog box opens.

2. On the **Security** page, click the **Add** button, and then select the logon account for either the JP1/Script service or the Script Launcher service.

In the Select Users or Groups dialog box that opens, select the logon account for each service.

3. On the **Security** page, select the users you added, and then specify the following information for **Permissions**:

- Read and execution: Permitted
- Folder contents listing: Permitted
- Read: Permitted

4. Make sure that unnecessary users are not set.

Delete any unnecessary users.

9. Stop the JP1/Script service and Script Launcher service.

10. Start the JP1/Script service and Script Launcher service.

Make sure that the following messages are output to the event log.

For the JP1/Script service:

```
Starting the NetExec thread (NetExec command execution users restricted)
of the JP1/Script service.
```

For the Script Launcher service:

```
Starting the NetExec thread (NetExec command execution users restricted)
of the Script Launcher service.
```

Important note

After you change the users permitted to remotely execute the `NetExec` command, restart the JP1/Script service or the Script Launcher service. In addition, when the permission is set for a group in the `NetExec` command restriction policy file, if you add or delete users or groups belonging to the permitted group, the added or deleted

users or groups might not be properly granted or denied permission after the JP1/Script service or Script Launcher service restarts. If this happens, restart Windows.

3

JP1/Script Operations

This chapter describes how to use JP1/Script Manager, Editor, Easy Input, Trace Viewer, Trace files, Menu Editor, Process Viewer, and Execution Environment File Converter.

3.1 Manager operations

You can perform the following operations in the Script Manager window:

- Choose an editor
- Create and save a script
- Create a script using the Easy Input facility
- Edit a script
- Check script syntax
- Copy a script
- Add a script
- Delete a script
- Rename a script
- Set the script execution environment (all items)
- Set the script execution environment (individual items)
- Set a script to start automatically
- Execute a script
- Create a menu form
- Quit Manager
- Change a folder[#]
- Show or hide the toolbar[#]
- Show or hide the status bar[#]
- View script files as large or small icons[#]
- View script files as a list[#]
- Refresh the client area[#]

#

These operations are not described. They are either standard Windows operations or simply involve choosing a command from a menu.

3.1.1 Script Manager window and menus

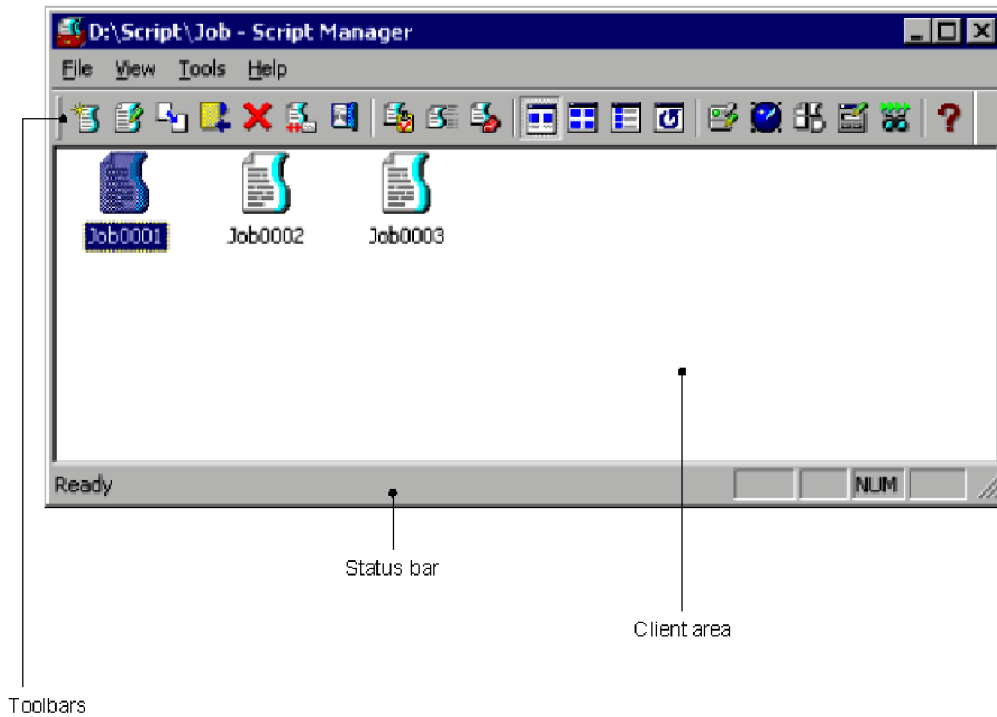
To create a script using JP1/Script, you begin by starting Manager. When you double-click the Manager icon, the Script Manager window appears. In this window you can perform a variety of script operations, such as checking the syntax and saving a created script.

This section describes the functions and menus of the Script Manager window that appears when you start Manager.

(1) Script Manager window

Figure 3-1 shows the Script Manager window and the names of its components.

Figure 3–1: Script Manager window



(a) Toolbar

The toolbar contains buttons representing the most frequently used commands on the pull-down menus. By simply clicking a button, you can execute the corresponding command. You can hide the toolbar by toggling **Toolbar** in the **View** menu.

The following command buttons appear in the toolbar of the Script Manager window:

Create

Creates a new script file.

Edit

Edits a script file.

Copy

Copies a script file.

Add

Adds a script file from another folder to the current folder.

Delete

Deletes a script file.

Rename

Renames a script file.

Exit

Quits Manager.

Syntax Check

Checks the syntax of a script file.

Execute

Executes a script file.

Set Execution Environment

Sets the environment for executing a script file.

Large Icons

Displays script files as large icons.

Small Icons

Displays script files as small icons.

List

Displays script files as a list.

Refresh

Updates the information displayed in the client area.

Set Automatic Start

Sets a script file to start automatically.

Trace Viewer

Starts Trace Viewer.

Easy Input

Starts Easy Input facility.

Menu Editor

Starts Menu Editor.

Process Viewer

Starts Process Viewer.

Help

Displays the JP1/Script online help.

(b) Status bar

The status bar displays messages about the processing being carried out by Manager and status messages at completion of processing.

(c) Client area

The client area displays icons that represent files. You can drag and rearrange the icons.

A scroll bar appears if the icons do not all fit within the visible client area.

(2) Menus in the Script Manager window

The pull-down and pop-up menus in the Script Manager window are described next.

(a) Menus and commands

Table 3-1 lists the Script Manager commands (functions) provided in the pull-down menus.

Table 3–1: Script Manager menus and commands

Menu	Command (function)	Description
File	Create	Starts the editor for creating a new script file.
	Edit	Starts the editor for editing an existing script file.

Menu	Command (function)	Description
File	Copy	Copies a script file from the current folder to another folder.
	Add	Adds a script file from another folder to the current folder.
	Delete	Deletes a created script file.
	Rename	Renames a script file.
	Syntax Check	Checks the syntax of a created script file.
	Execute	Executes a script file.
	Set Execution Environment - All Items	Sets the execution environment for a created script file.
	Set Execution Environment - Command Line	Changes only the command line in the execution environment for a created script file.
	Set Execution Environment - Working Folder	Changes only the working folder in the execution environment for a created script file.
	Set Execution Environment - Start Date	Changes only the start date in the execution environment for a created script file.
	Set Execution Environment - Terminate Time	Changes only the termination time in the execution environment for a created script file.
	Set Execution Environment - Trace Output Folder	Changes only the trace output folder in the execution environment for a created script file.
	Set Execution Environment - User Trace Information	Changes only the user trace information in the execution environment for a created script file.
	Change Folder	Changes the folder displayed in the client area.
	(Folder name)	Displays the previous folder name.
Exit	Terminates the Menu Manager of the Script Manager window.	
View	Toolbar	Shows or hides the toolbar.
	Status Bar	Shows or hides the status bar.
	Large Icons	Displays the created script files as large icons.
	Small Icons	Displays the created script files as small icons.
	List	Displays the created script files as a list.
	Refresh	Updates the information displayed in the client area.
Tools	Set Automatic Start	Sets a script file to start automatically.
	Link Editor	Links an editor for editing a script file.
	Start Trace Viewer	Starts Trace Viewer.
	Start Easy Input	Starts the Easy Input facility.
	Start Menu Editor	Starts Menu Editor.
	Start Process Viewer	Starts Process Viewer.
	Options	Performs the following functions: <ul style="list-style-type: none"> • Enables or disables command execution from clients. • Displays access permissions for a server. • Specifies the Script Engine version to be used.

Menu	Command (function)	Description
Help	Contents	Displays the contents of the JP1/Script online help.
	Search by Keyword	Lists the keywords of the JP1/Script online help.
	About JP1/Script	Displays version information for JP1/Script.

(b) Pop-up menu

To display the pop-up menu in the Script Manager window, right-click in the client area. Table 3-2 lists the commands in the Script Manager pop-up menu.

Table 3–2: Commands in the Script Manager pop-up menu

Command (function)	Description
Create	Creates a new script file.
Edit	Edits a script file.
Copy	Copies a script file from the current folder to another folder.
Add	Adds a script file from another folder to the current folder.
Delete	Deletes a created script file.
Rename	Renames a script file.
Syntax Check	Checks the syntax of a created script file.
Execute	Executes a script file.
Set Execution Environment - All Items	Sets the execution environment for a created script file.
Set Execution Environment - Command Line	Changes only the command line in the execution environment for a created script file.
Set Execution Environment - Working Folder	Changes only the working folder in the execution environment for a created script file.
Set Execution Environment - Start Date	Changes only the start date in the execution environment for a created script file.
Set Execution Environment - Terminate Time	Changes only the termination time in the execution environment for a created script file.
Set Execution Environment - Trace Output Folder	Changes only the trace output folder in the execution environment for a created script file.
Set Execution Environment - User Trace Information	Changes only the user trace information in the execution environment for a created script file.
Start Menu Editor	Starts Menu Editor.
Properties	Displays information about a script file.

3.1.2 Mouse and key operations in the Script Manager window

(1) Mouse operations

The following describes mouse operations in the client area of the Script Manager window. Table 3-3 lists mouse operations on icons, and Table 3-4 lists mouse operations on the background of the client area.

Table 3–3: Mouse operations on icons in the Script Manager window

Operation	Processing
Click	Clears the previous selection and makes a new selection.
Double-click	Executes a script.
Right-click	Displays a pop-up menu.

Table 3–4: Mouse operations on the background of the Script Manager window

Operation	Processing
Click	Clears the previous selection.
Drag	Selects the icons in the indicated area.
Right-click	Displays a pop-up menu.

(2) Key operations

Table 3-5 lists key operations in the client area of the Script Manager window.

Table 3–5: Key operations in the Script Manager window

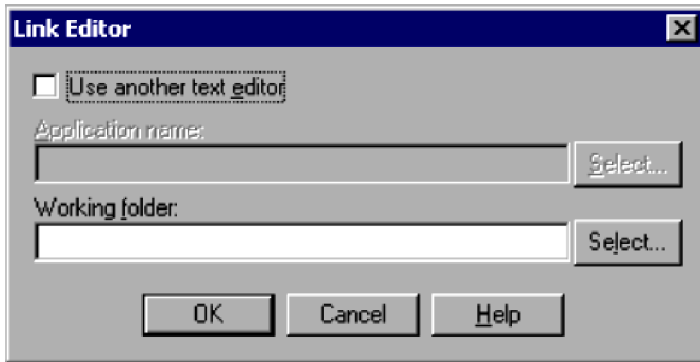
Operation	Processing
Ctrl + N	Creates a new script file.
Ctrl + O	Edits the selected script file.
DEL	Deletes the selected script file.
Enter	Executes the selected script file.
Shift + Enter	Checks the syntax of the selected script file.
Shift + Ctrl + Enter	Sets the execution environment of the selected script file.

3.1.3 Choosing an editor

JP1/Script provides a dedicated editor (Editor) for creating scripts. This editor is set as the default. However, you can use a different editor to create a script if you wish.

To set a different editor:

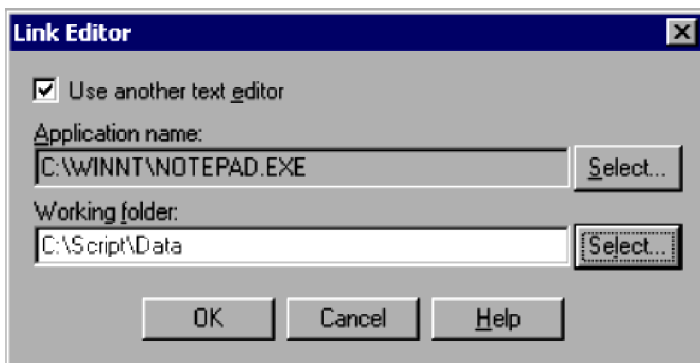
1. From the **Tools** menu, choose **Link Editor**.
The Link Editor dialog box appears.



For details about using this dialog box, see [4.1.21 Link Editor dialog box](#).

2. To use an editor other than Editor, select the **Use another text editor** check box, click the **Select** button, and then select the editor you want to associate. If necessary, specify a working folder for the editor too.

The following example links Notepad as the editor for script creation.



To cancel the specified editor and go back to using Editor, clear the **Use another text editor** check box.

3. Click the **OK** button.

The editor is now linked.

Notes

- You cannot use the standard Windows WordPad. Set another type of program that supports saving in text format.
- If you do not specify a working folder at step 2, the editor's folder is assumed.
- If you click **Cancel** at step 3, editor linkage is canceled.

3.1.4 Creating and saving a script

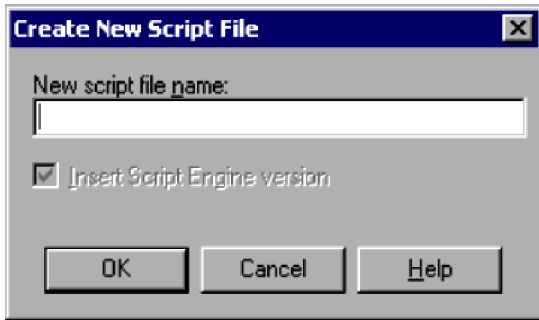
The following describes how to create a script file.

Before you begin this task, you must set the editor to be used. For details, see [3.1.3 Choosing an editor](#).

To create and save a script file:

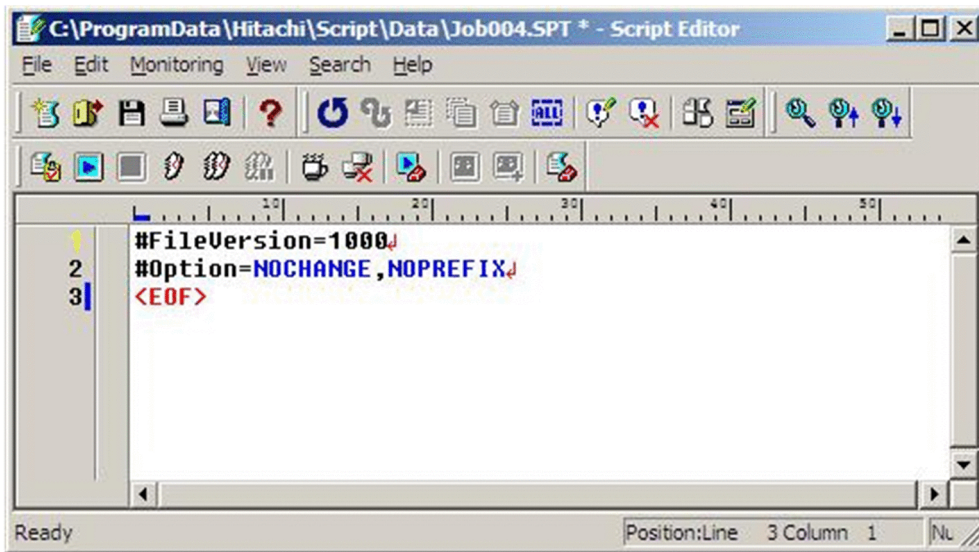
1. Choose **File, Create**.

The Create New Script File dialog box appears.

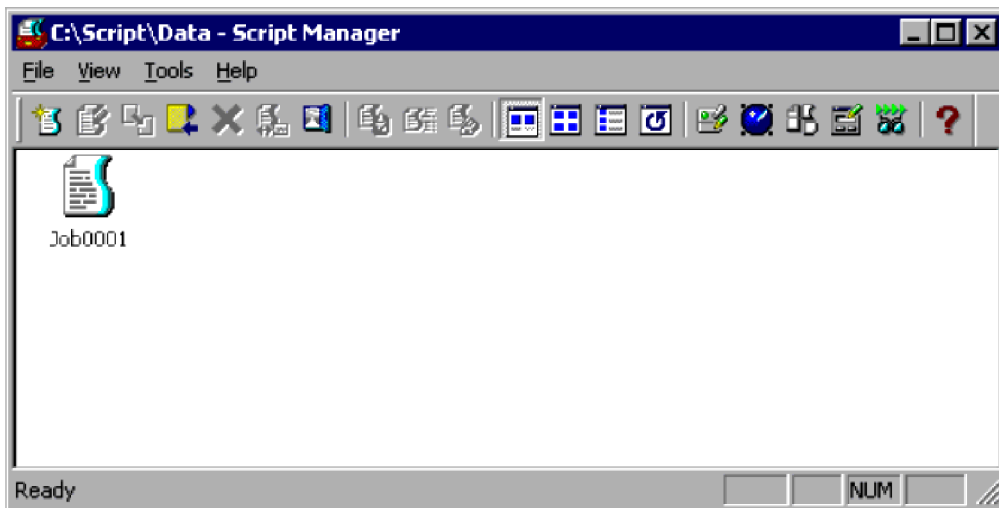


For details about using this dialog box, see [4.1.1 Create New Script File dialog box](#).

2. Type the name of the script file to be created, omitting the extension, and then click **OK**. The linked editor starts.



3. Create and save the script, then quit the editor. An icon representing the created script file appears in the client area.



Notes

- If you used an editor other than Editor, choose **View**, and then **Refresh** to display the created script file in the client area.

- You cannot use the standard Windows WordPad. If WordPad is set as the editor, change the setting as described in [3.1.3 Choosing an editor](#).
- Omit the file extension when you enter the script file name at step 2. JP1/Script automatically assigns the extension `.SPT`.
- Manager and the editor work independently. If you exit Manager after starting the editor, the editor does not terminate.

3.1.5 Creating a script with Easy Input

You can use the Easy Input program to enter commands and statements in a script file. With Easy Input, you can enter commands easily and accurately without needing any command-specific knowledge.

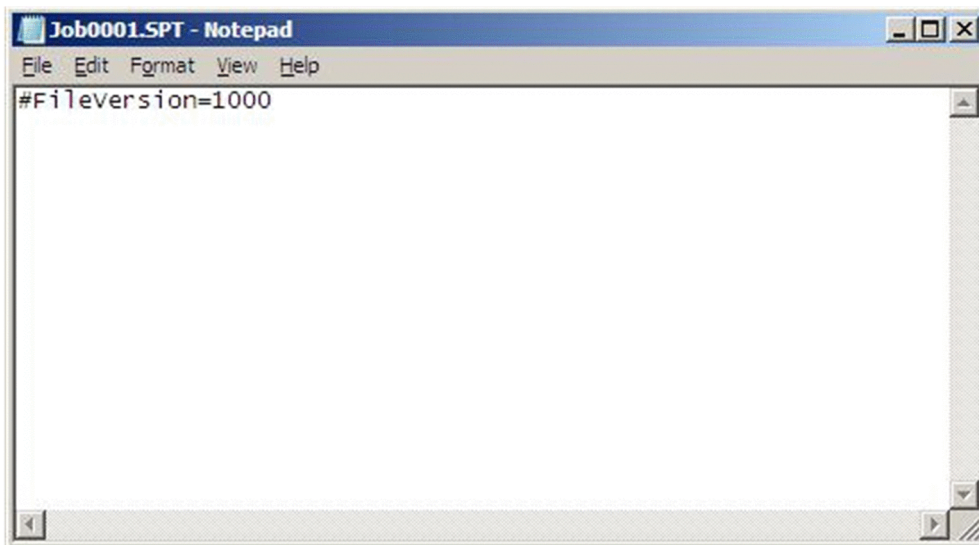
Entered commands and statements can be pasted via the clipboard to the editor you are using.

The Easy Input procedure depends on the associated editor; that is, whether you use Editor or another editor.

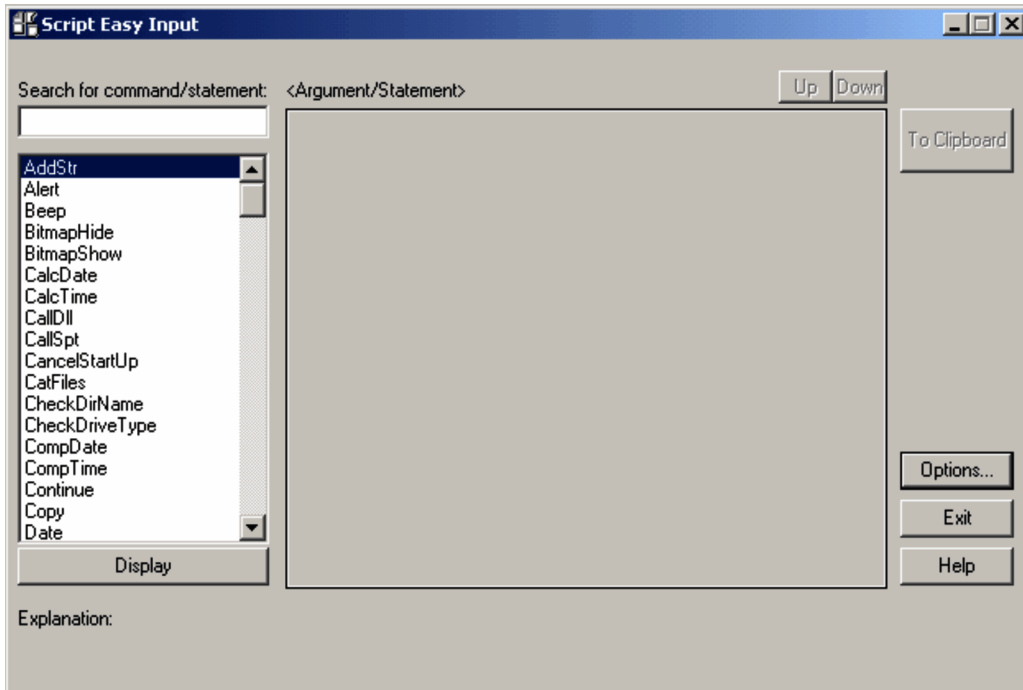
- If the dedicated Editor has been associated
See [3.2.4 Using Easy Input](#).
- If you are using any other editor: As below.

To create a script with Easy Input using an editor other than Script Editor:

1. Choose **File, Create**. Alternatively, select a script file and choose **File, Edit** to start the editor.
The linked editor starts (in the following example, the linked editor is Notepad).

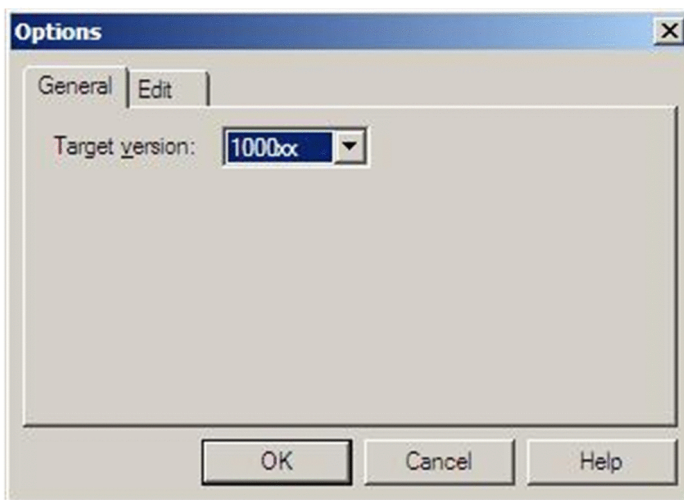


2. Choose **Tools, Start Easy Input**. Alternatively, from the Windows **Start** menu, choose **Programs** and then **JP1_Script, Easy Input** to start the Easy Input facility.
The Easy Input window appears.



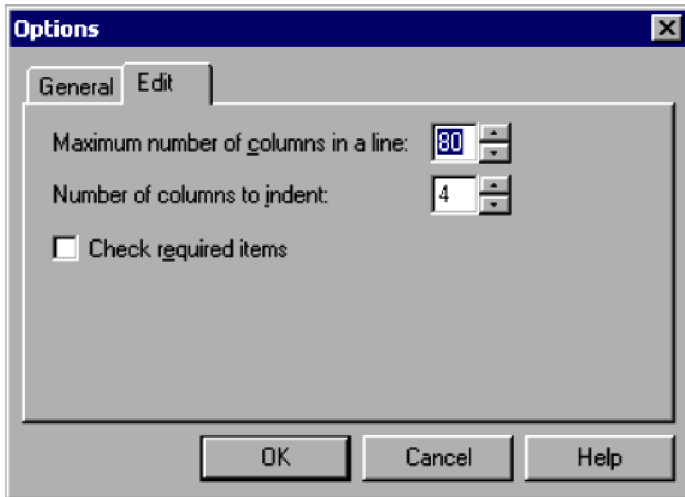
For details about using this window, see [3.3 Easy Input operations](#).

3. Click **Options**. In the **General** tab of the Options dialog box, set the version of the script file to be created.



For details about using this dialog box, see [3.3.3 Specifying the JPI/Script version for script creation](#).

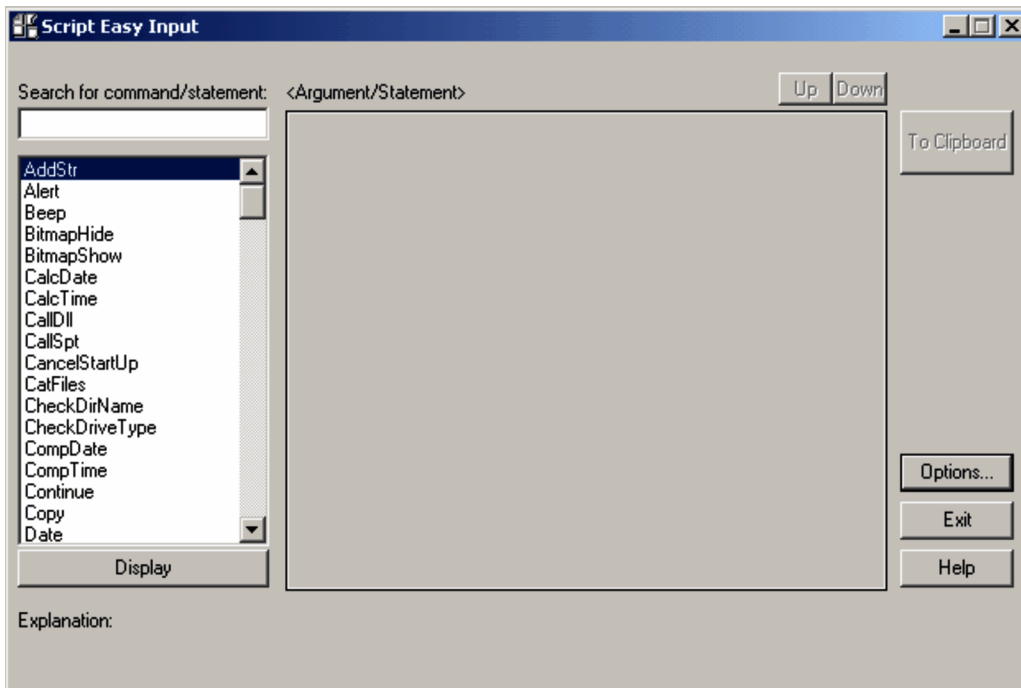
4. Click the **Edit** tab. In the **Edit** tab of the Options dialog box, set the format for output to the clipboard.



For details about using this dialog box, see [3.3.4 Setting the line length and indent size](#).

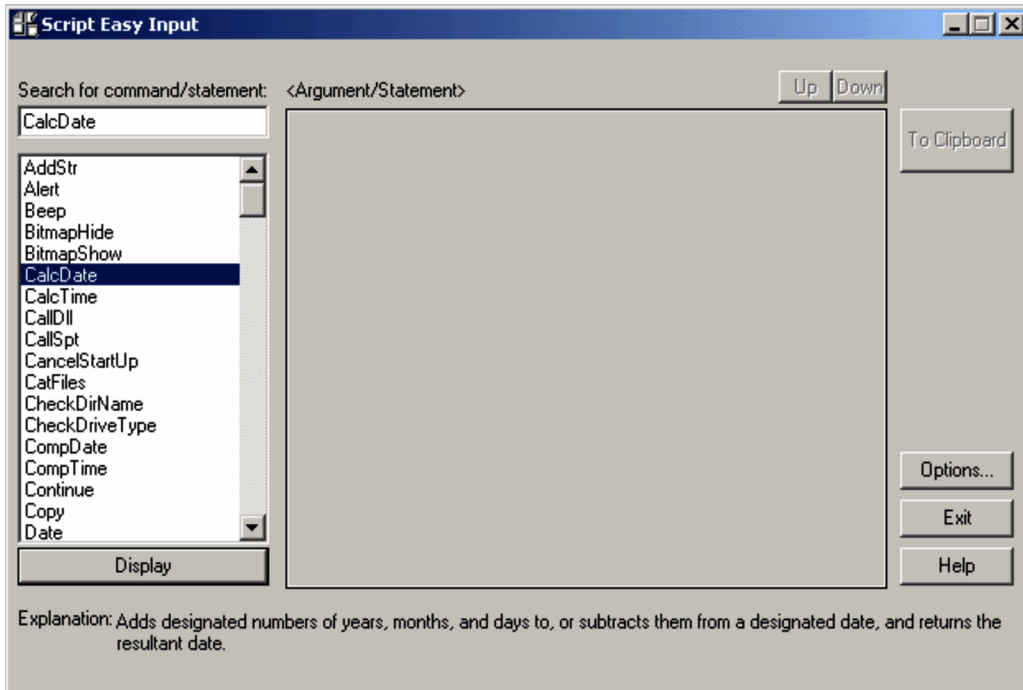
5. Click the **OK** button.

The script file version and clipboard output format are set. The Script Easy Input window appears again.

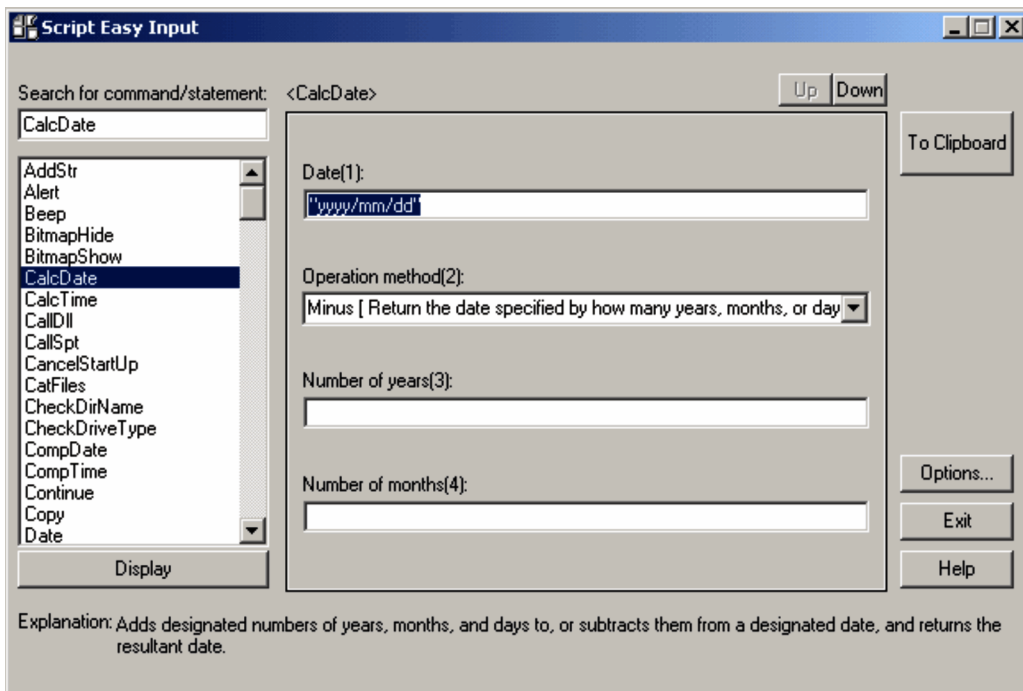


6. Find and select the command or statement you want to use.

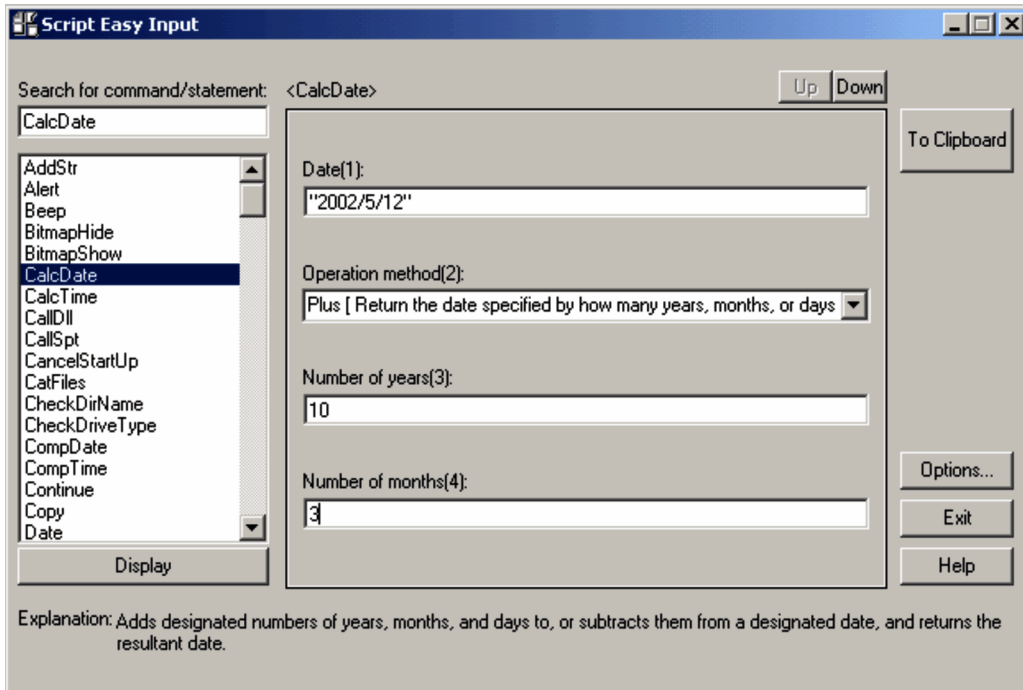
The focus shifts to the command or statement you selected.



- Click the **Display** button, or double-click the command or statement.
If you selected a command, the arguments that need to be set appear in the **Argument/Statement** area.
If you selected a statement, the statement syntax appears in this area.



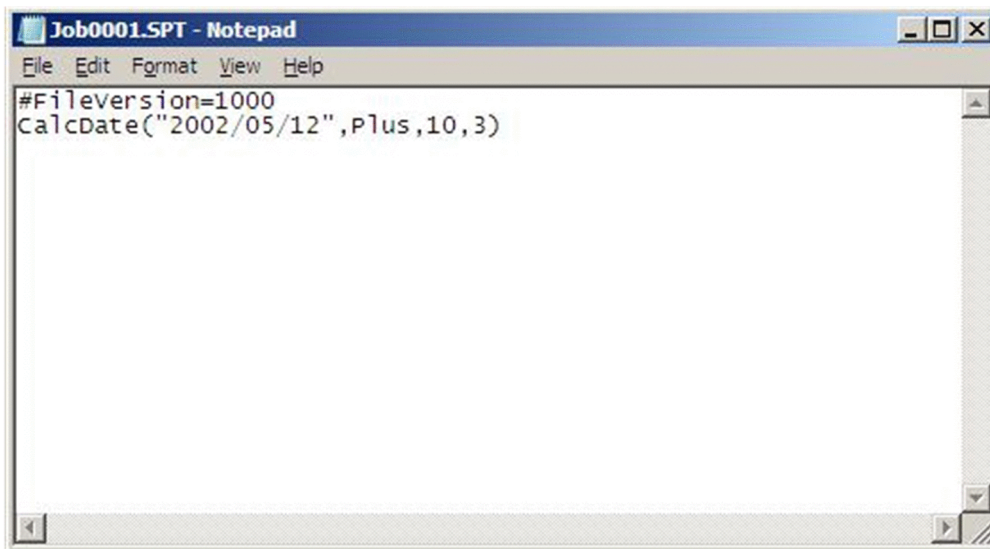
- Enter the command arguments.



9. Click the **To Clipboard** button.

The contents that you set in the **Argument/Statement** area are pasted to the clipboard.

10. Paste the completed command or statement from the clipboard to the editor.



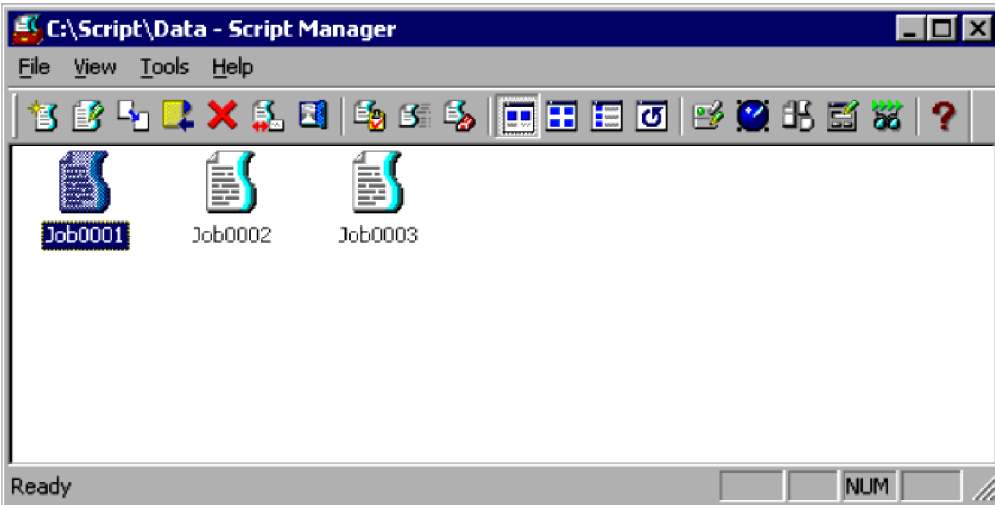
Notes

- Select a command name or statement name in the **Search for command/statement** list box and then right-click to view help about that command or statement.
- If the command you selected in step 7 has no arguments, a message to that effect appears in the **Argument/Statement** area.

3.1.6 Editing a script

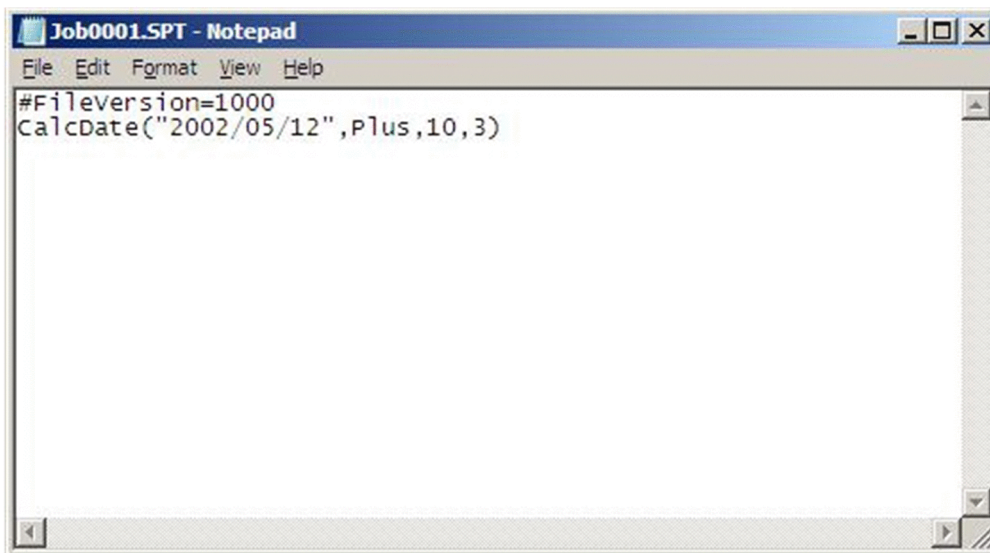
To edit a created script:

1. In the client area, select the icon of the script file you want to edit.



2. Choose **File, Edit**.

The linked editor starts.



3. Edit and save the script, then quit the editor.

Notes

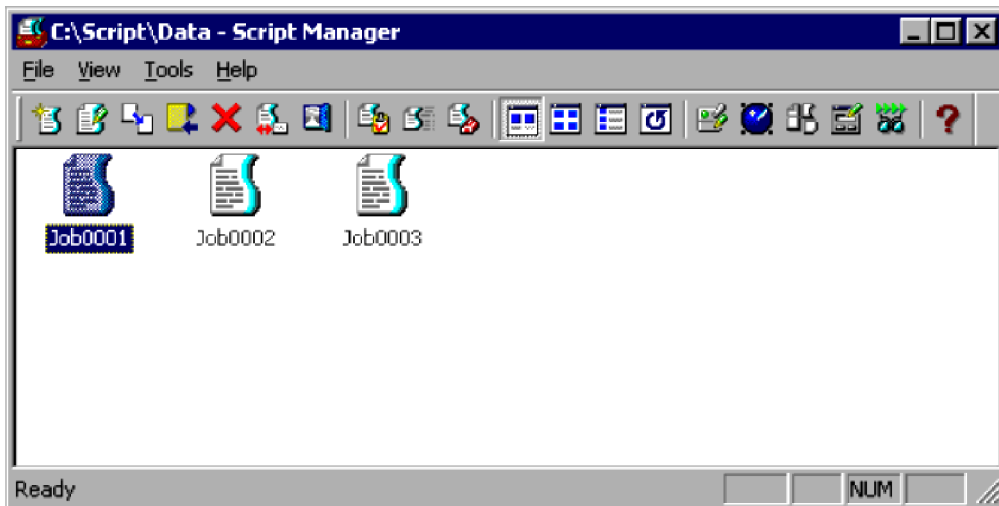
- You cannot use the standard Windows WordPad. If WordPad is set as the linked editor, change the setting as described in [3.1.3 Choosing an editor](#).
- The default is that Editor is associated as the editor for editing scripts.

3.1.7 Checking script syntax

After completing a script, check its syntax. The syntax check includes lexical and grammatical analysis only. No commands are actually executed.

To check script syntax:

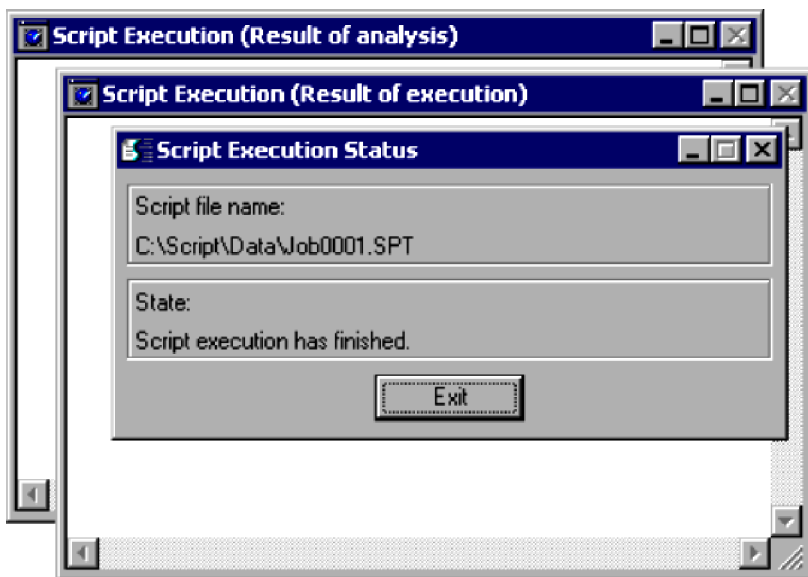
1. In the client area, select the icon of the script file you want to check.



2. Choose **File, Syntax Check**.

While the syntax check is being performed, windows show the execution status, execution trace, and analysis trace. The syntax check result is displayed as an analysis trace.

At completion of the syntax check, the **Exit** button is enabled in the window showing the execution status.



3. Check the contents of the analysis trace.

You can edit the script while viewing the analysis results. Start the editor by choosing **File, Edit**.

4. Click the **Exit** button in the window displaying the execution status.

The three windows close.

Note

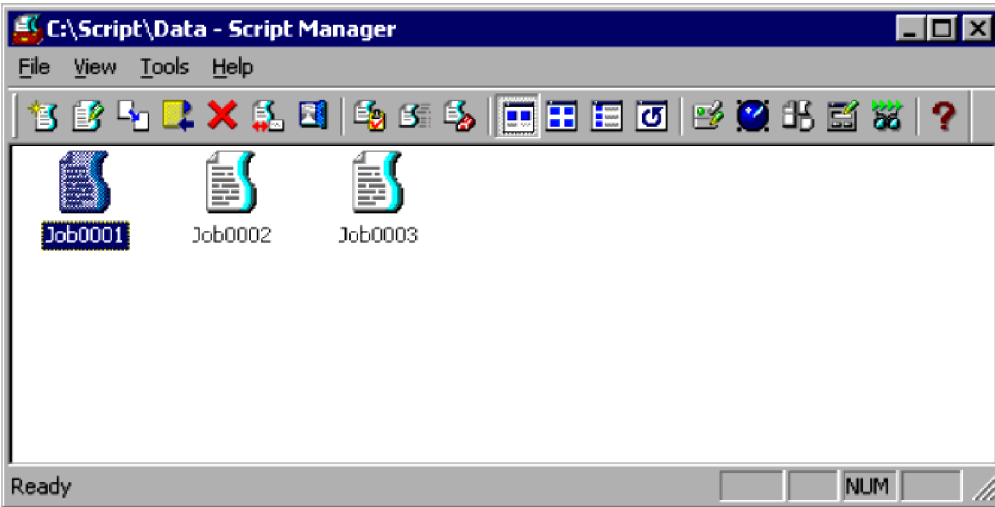
When you perform a syntax check, JP1/Script creates an analysis trace file. To print this file, start Trace Viewer, select the trace file, and then choose **File, Print**. For details, see [3.4.10 Printing a trace file](#).

3.1.8 Copying a script

You can copy script files from the current folder to another folder.

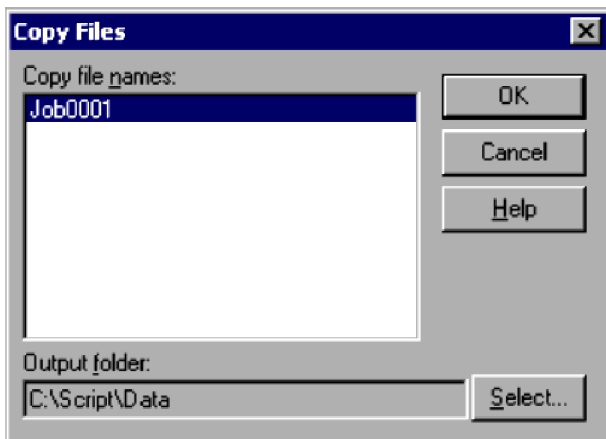
To copy a script:

1. In the client area, select one or more icons representing the script file(s) you want to copy.



2. Choose **File, Copy**.

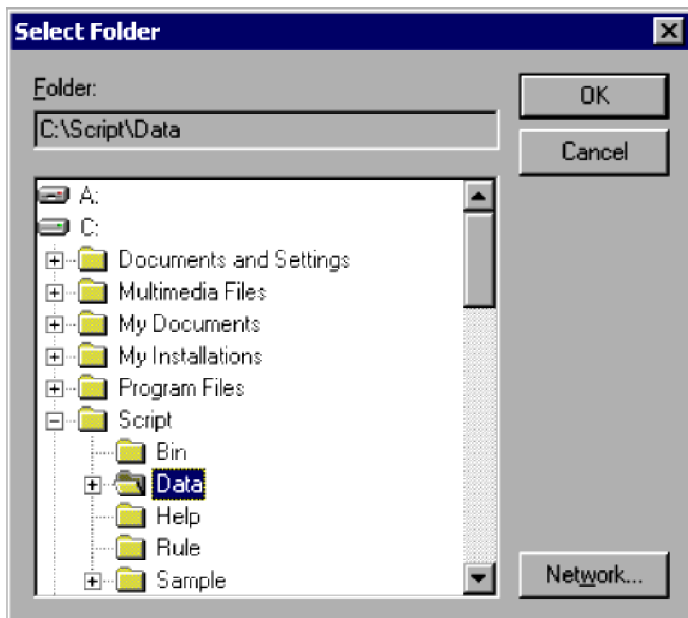
The Copy Files dialog box appears.



For details about using this dialog box, see [4.1.2 Copy Files dialog box](#).

3. Click the **Select** button.

The Select Folder dialog box appears. Select a folder to copy the file(s) to.



For details about using this dialog box, see [4.1.29 Select Folder dialog box](#).

4. Click the **OK** button.

The file(s) are copied.

Notes

- You can cancel copying of one or more files when the Copy Files dialog box is open. Just click the file name in the dialog box. This cancels the selection and excludes the file from the copy operation.
- If you click **Cancel** at step 4, file copying is canceled and the dialog box closes.
- If a file of the same name already exists in the copy destination folder, the Confirm File Overwrite dialog box appears.

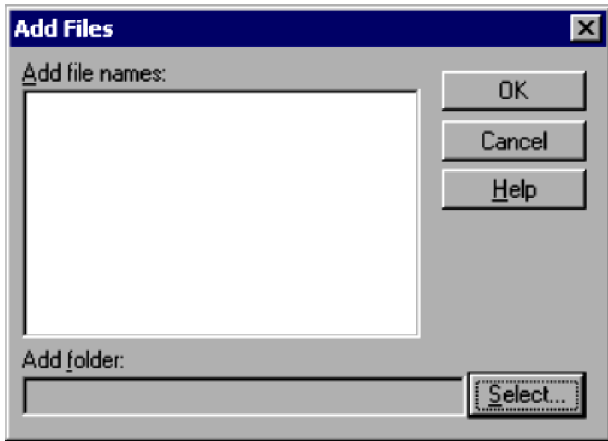
3.1.9 Adding a script

You can add script files from another folder to the current folder.

To add a script:

1. Choose **File, Add**.

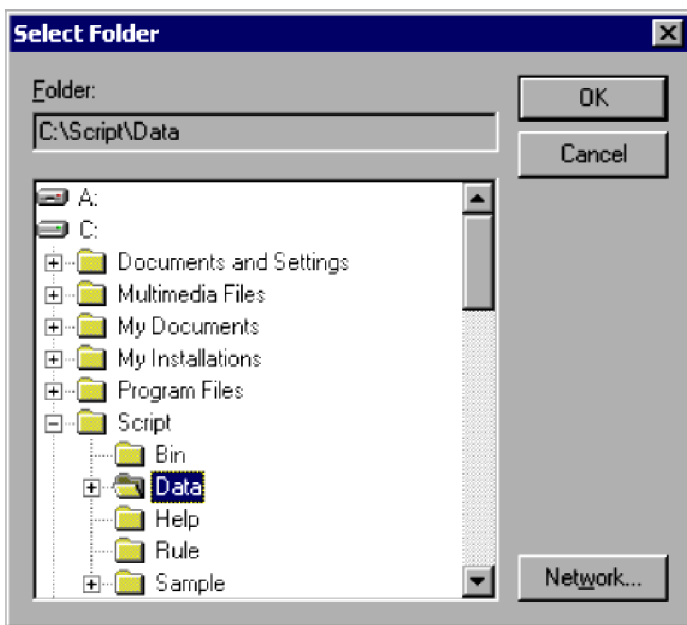
The Add Files dialog box appears.



For details about using this dialog box, see [4.1.3 Add Files dialog box](#).

2. Click the **Select** button.

The Select Folder dialog box appears. Select a folder from which to add the file(s).



For details about using this dialog box, see [4.1.29 Select Folder dialog box](#).

3. Select one or more files to add.

4. Click the **OK** button.

The file(s) are added.

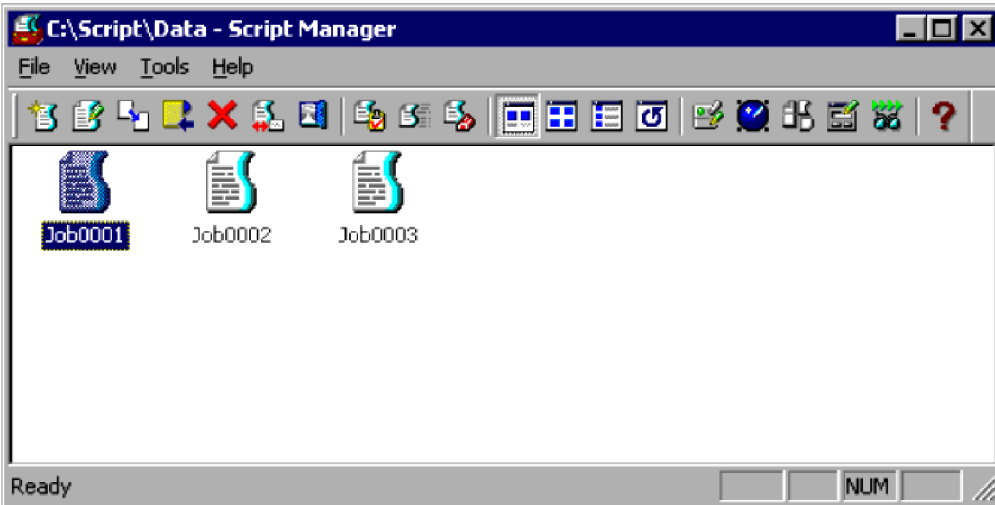
Notes

- If you click **Cancel** at step 4, file addition is canceled and the dialog box closes.
- If a file of the same name already exists in the current folder, the Confirm File Overwrite dialog box appears.

3.1.10 Deleting a script

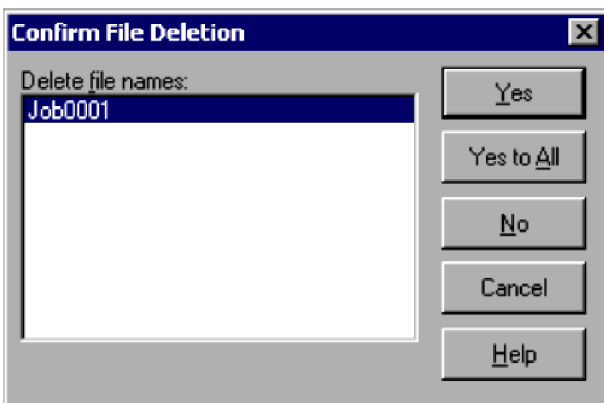
To delete a created script file:

1. In the client area, select one or more icons representing the script file(s) you want to delete.



2. Choose **File, Delete**.

The Confirm File Deletion dialog box appears.



For details about using this dialog box, see [4.1.5 Confirm File Deletion dialog box](#).

3. Click the **Yes** button.

The first script file is deleted. You are then asked to confirm deletion of the next file you selected. This process is repeated for all the files you selected. When all the files have been deleted, the dialog box closes.

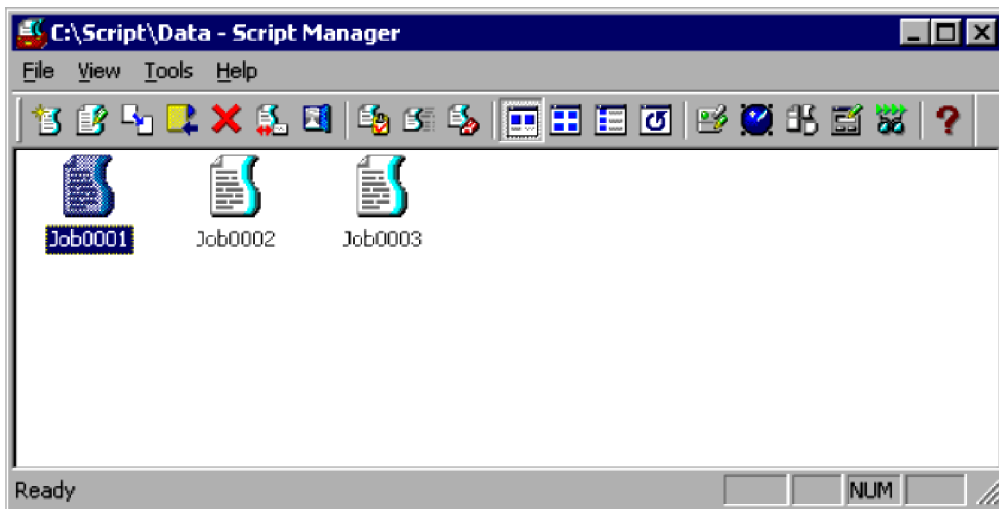
Notes

- If you click **Yes to All** at step 3, all the selected files are deleted.
- If you click **Cancel** at step 4, file deletion is canceled and the dialog box closes.
- You can cancel deletion of one or more files when the Confirm File Deletion dialog box is open. Just click the file name in the dialog box. This cancels the selection and excludes the file from deletion.

3.1.11 Renaming a script

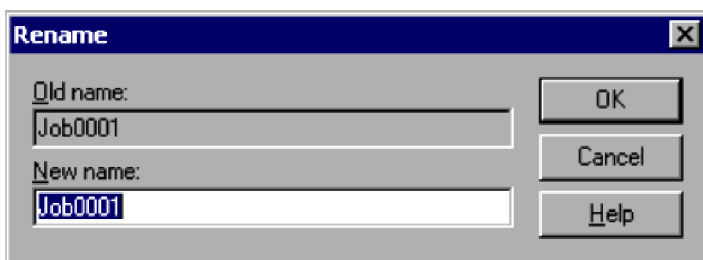
To rename a selected script file:

1. In the client area, select the icon of the script file you want to rename.



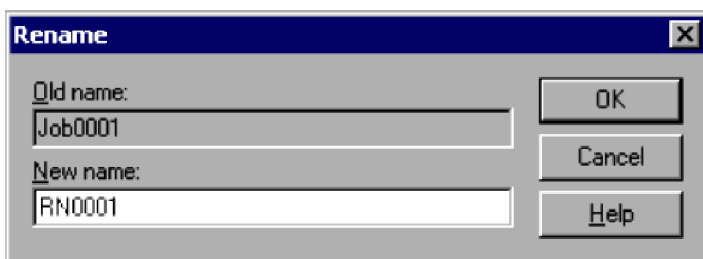
2. Choose **File, Rename**.

The Rename dialog box appears.



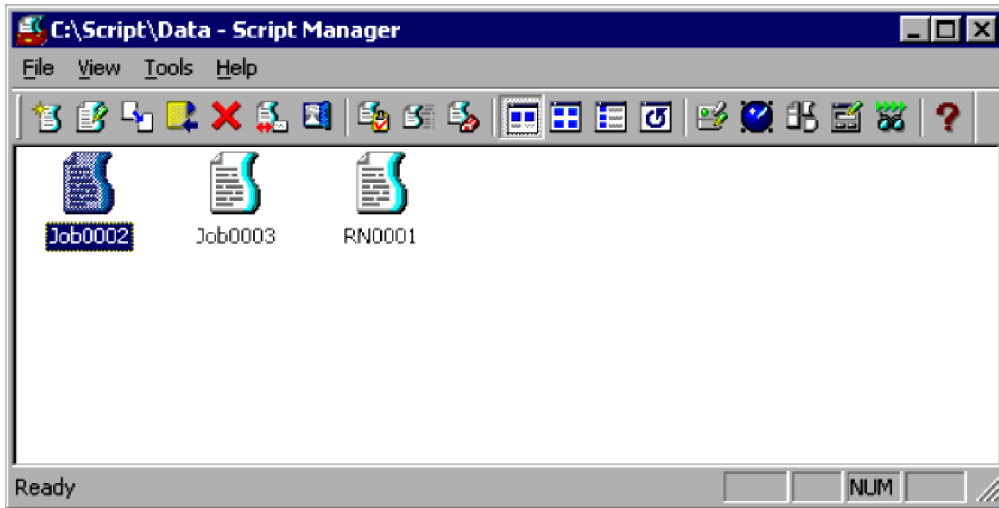
For details about using this dialog box, see [4.1.6 Rename dialog box](#).

3. Enter the new name.



4. Click the **OK** button.

The script file is renamed and the dialog box closes.



Note

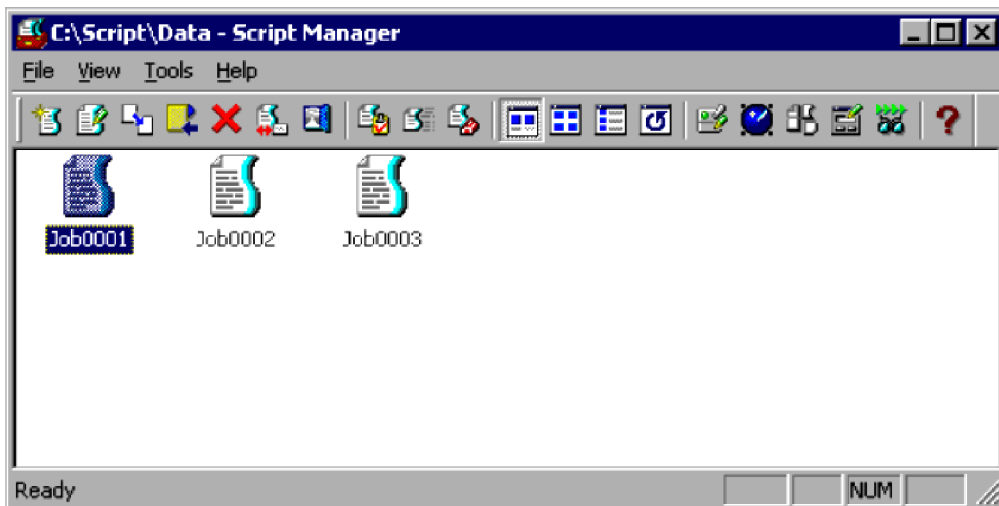
If you click **Cancel** at step 4, file renaming is canceled and the dialog box closes.

3.1.12 Setting the script execution environment (all items)

The following describes how to set the execution environment for a created script file. You can also do this from Editor. For details, see [3.2.13 Setting the script execution environment](#).

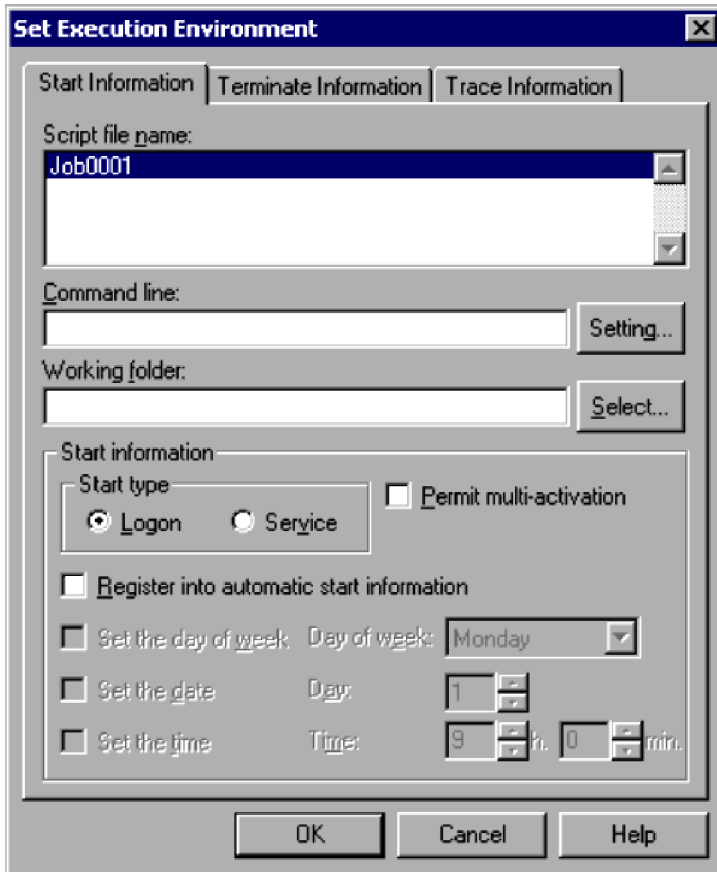
To set the script execution environment from the Script Manager window:

1. In the client area, select one or more icons representing the script file(s) for which you want to set the execution environment.



2. Choose **File, Set Execution Environment**, then **All Items**.

The Set Execution Environment (Start Information) dialog box appears.



For details about using this dialog box, see [4.1.8 Set Execution Environment \(Start Information\) dialog box](#).

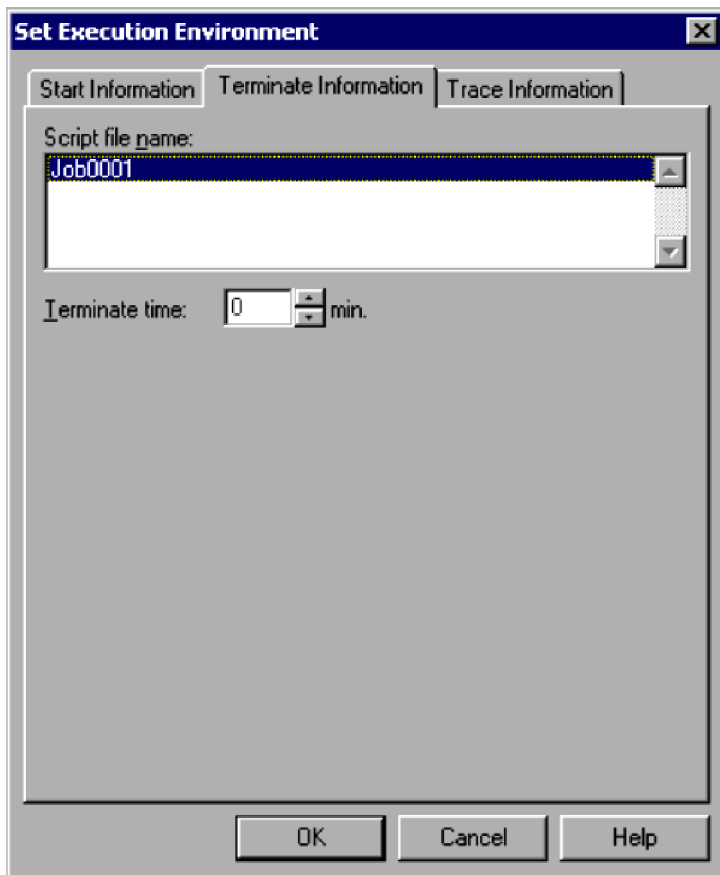
3. Set the required start information.

You must write a command line if a parameter is required for processing, or if you wish to set the output format of the analysis trace or execution trace.

For details on specifying command lines, see [6.2 Rules for writing command lines](#).

4. Click the **Terminate Information** tab.

The Set Execution Environment (Terminate Information) dialog box appears.



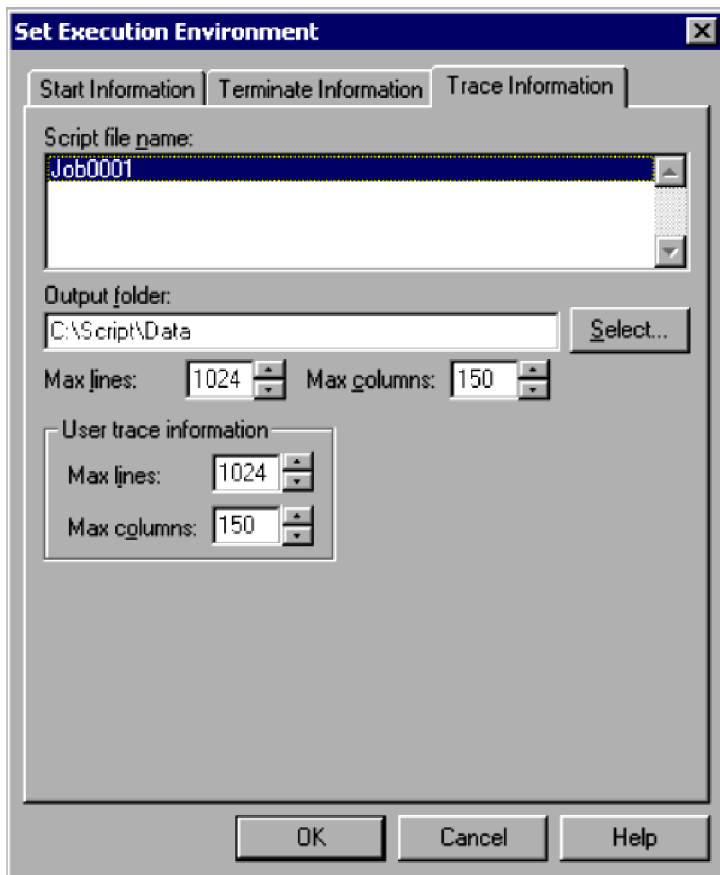
For details about using this dialog box, see [4.1.9 Set Execution Environment \(Terminate Information\) dialog box](#).

5. Set the termination time.

If you want to disable forced termination at a set time, make sure the time is set to zero.

6. Click the **Trace Information** tab.

The Set Execution Environment (Trace Information) dialog box appears.



For details about using this dialog box, see [4.1.10 Set Execution Environment \(Trace Information\) dialog box](#).

7. Set the required trace information.

You do not need to set any information in this dialog box unless you particularly want to set the number of output lines in the trace file or other such information.

8. Click the **OK** button.

The execution environment is set and the dialog box closes.

Notes

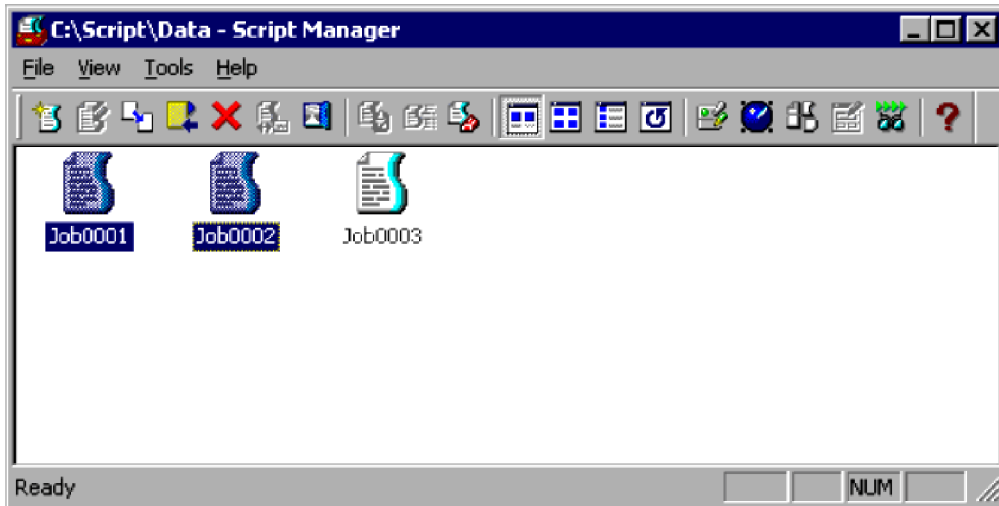
- If you attempt to set up a script execution environment while the JP1/Script service is not running, a dialog box containing the following error message appears:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.
- This setting is not enabled in the Script Launcher service.

3.1.13 Setting the script execution environment (individual items)

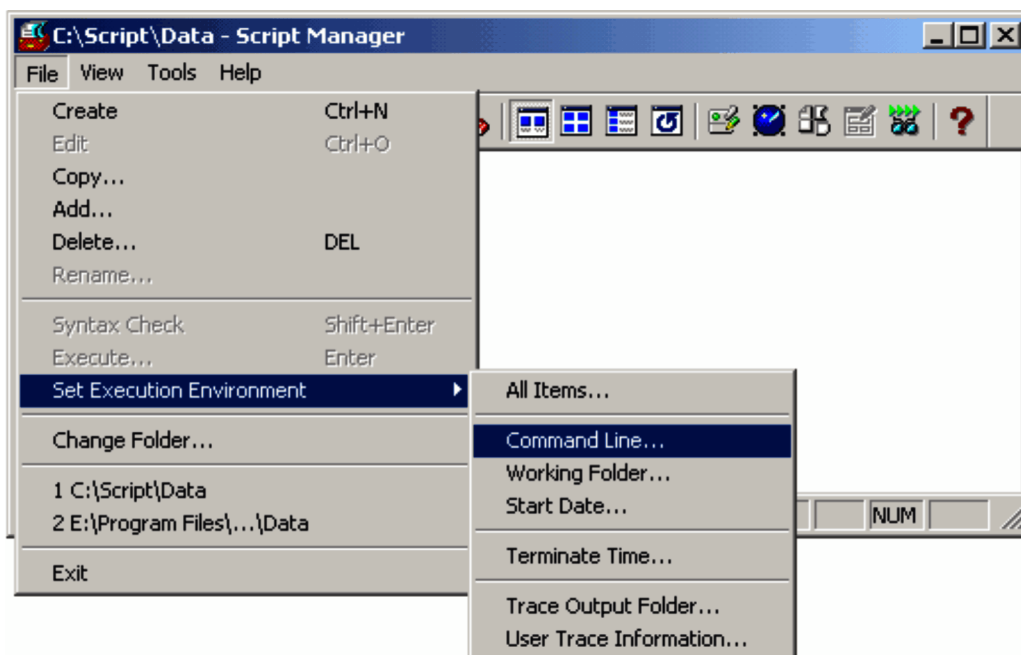
The following describes how to set individual items in the execution environment for a created script file.

To set the individual items in the script execution environment:

1. In the client area, select one or more icons representing the script file(s) for which you want to set the execution environment.

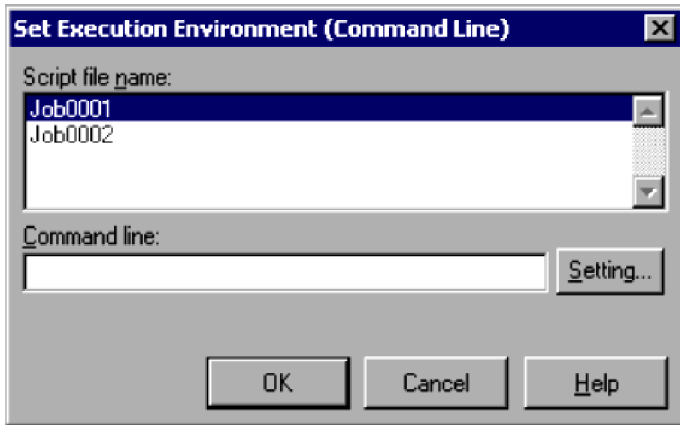


2. Choose **File**, **Set Execution Environment**, then choose **Command Line** or another command in the sub-menu.



Choose one of the following commands in the sub-menu:

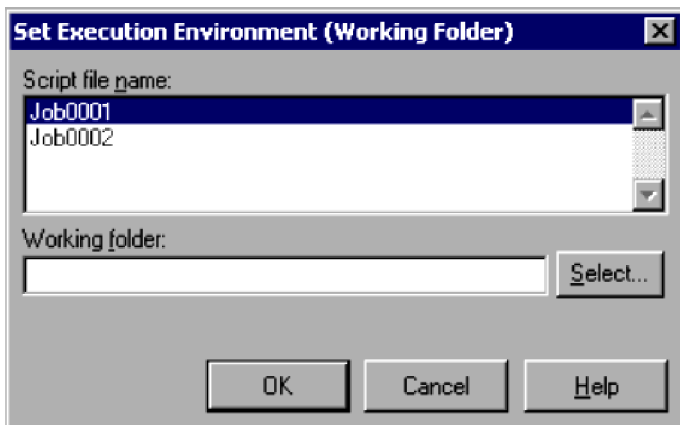
- **Command Line**
 - **Working Folder**
 - **Start Date**
 - **Terminate Time**
 - **Trace Output Folder**
 - **User Trace Information**
- If you choose **Command Line**:
The Set Execution Environment (Command Line) dialog box appears.



For details about using this dialog box, see [4.1.11 Set Execution Environment \(Command Line\) dialog box](#).

- If you choose **Working Folder**:

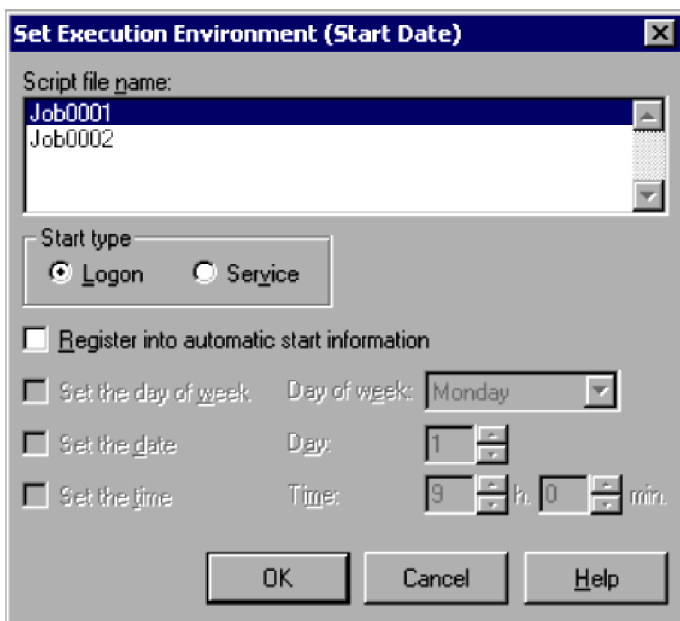
The Set Execution Environment (Working Folder) dialog box appears.



For details about using this dialog box, see [4.1.13 Set Execution Environment \(Working Folder\) dialog box](#).

- If you choose **Start Date**:

The Set Execution Environment (Start Date) dialog box appears.



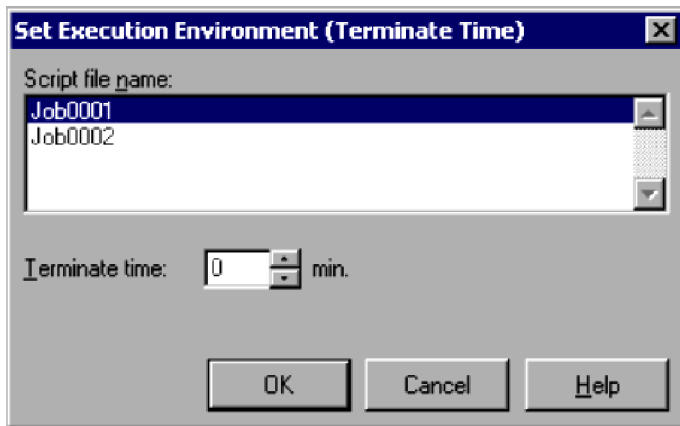
For details about using this dialog box, see [4.1.14 Set Execution Environment \(Start Date\) dialog box](#).

If you attempt to set up a script execution environment (start date) while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

- If you choose **Terminate Time**:

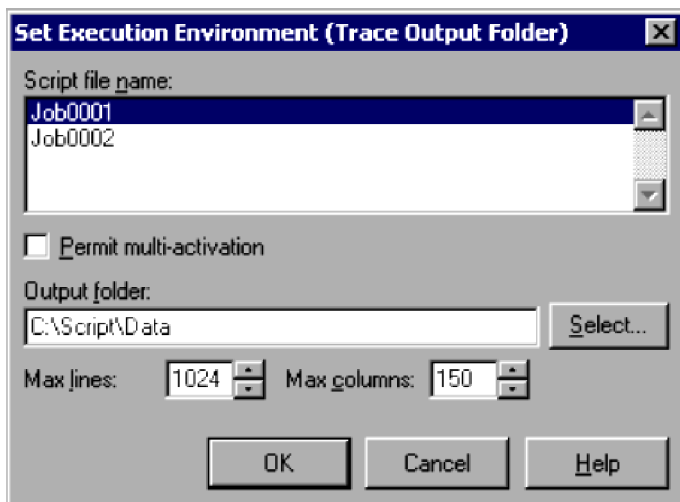
The Set Execution Environment (Terminate Time) dialog box appears.



For details about using this dialog box, see [4.1.15 Set Execution Environment \(Terminate Time\) dialog box](#).

- If you choose **Trace Output Folder**:

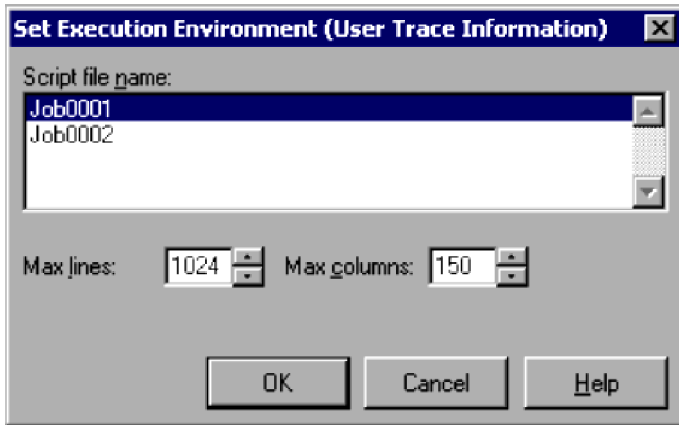
The Set Execution Environment (Trace Output Folder) dialog box appears.



For details about using this dialog box, see [4.1.16 Set Execution Environment \(Trace Output Folder\) dialog box](#).

- If you choose **User Trace Information**:

The Set Execution Environment (User Trace Information) dialog box appears.



For details about using this dialog box, see *4.1.17 Set Execution Environment (User Trace Information) dialog box*.

1. Set the required information.
2. Click the **OK** button.

The specific item is set in the execution environment and the dialog box closes.

Note

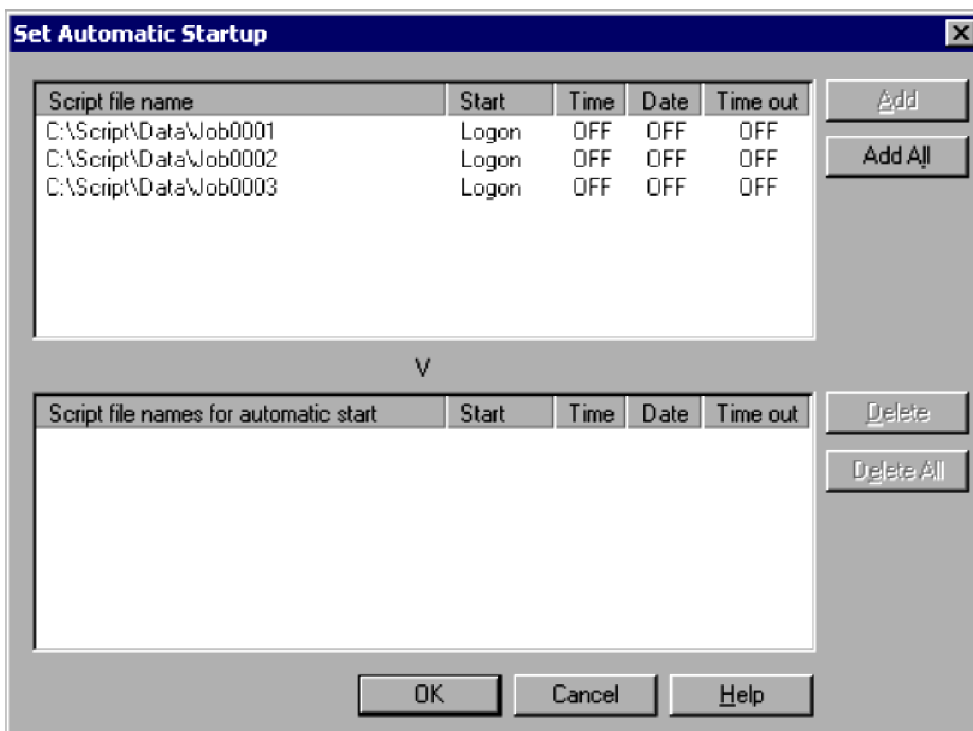
This setting is not enabled in the Script Launcher service.

3.1.14 Setting a script to start automatically

To set the name of a script file to be started automatically by the Script launcher or JP1/Script service:

1. Choose **Tools, Set Automatic Start**.

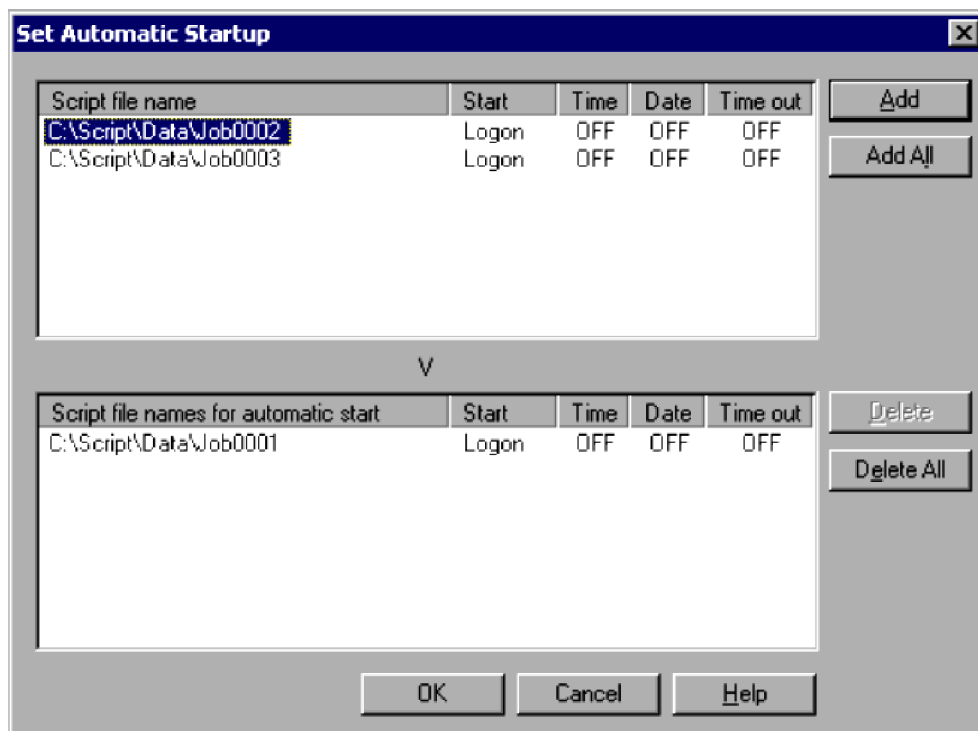
The Set Automatic Startup dialog box appears.



For details about using this dialog box, see *4.1.20 Set Automatic Startup dialog box*.

2. Select a script file name and then click **Add**. Alternatively, click the **Add All** button.

The selected script file name(s) are added to the **Script file names for automatic start** list box.



3. Click the **OK** button.

The contents of the **Script file names for automatic start** list box are exported as an automatic start information file.

- You can delete script file names from the **Script file names for automatic start** list box. Select one or more file names and then click **Delete**. Alternatively, click the **Delete All** button to delete all the file names in the list. The deleted script file names are added to the **Script file name** list box.
- If you click **Cancel** at step 3, the dialog box closes and the added files are ignored. The automatic start setting returns to initial status.

Notes

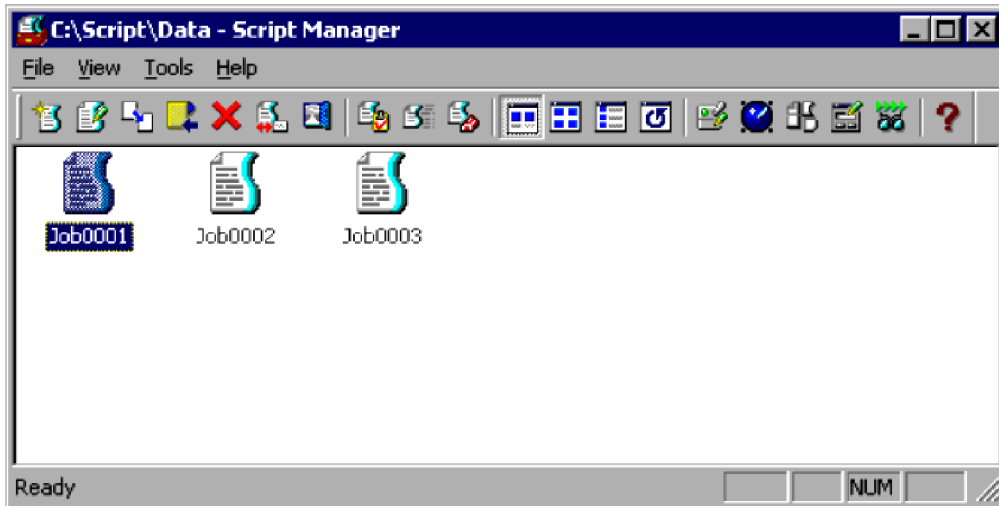
- If you attempt to set up a script to be started automatically while the JP1/Script service is not running, a dialog box containing the following error message appears:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.
- This setting is not enabled in the Script Launcher service.

3.1.15 Executing a script

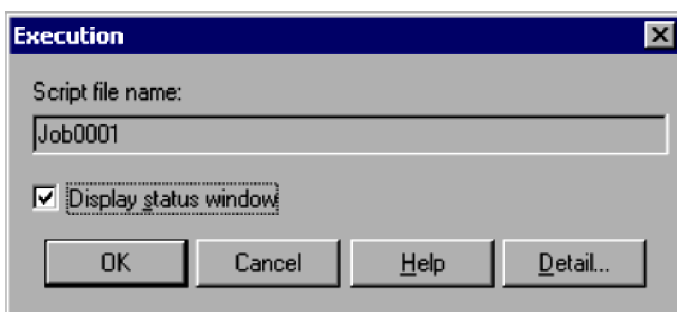
You can execute a created script file from the Script Manager window. This is normally performed to check the execution trace and script operation.

To execute a script from the Script Manager window:

1. In the client area, select the icon of the script file you want to execute.

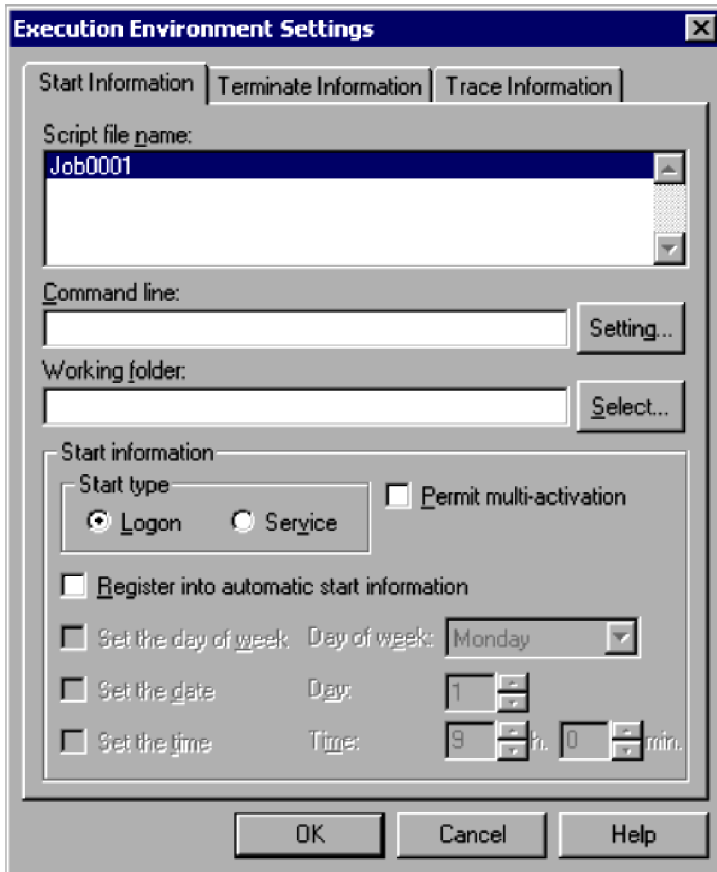


2. Choose **File, Execute**. Alternatively, double-click the file icon.
The Execution dialog box appears.



For details about using this dialog box, see [4.1.7 Execution dialog box](#).

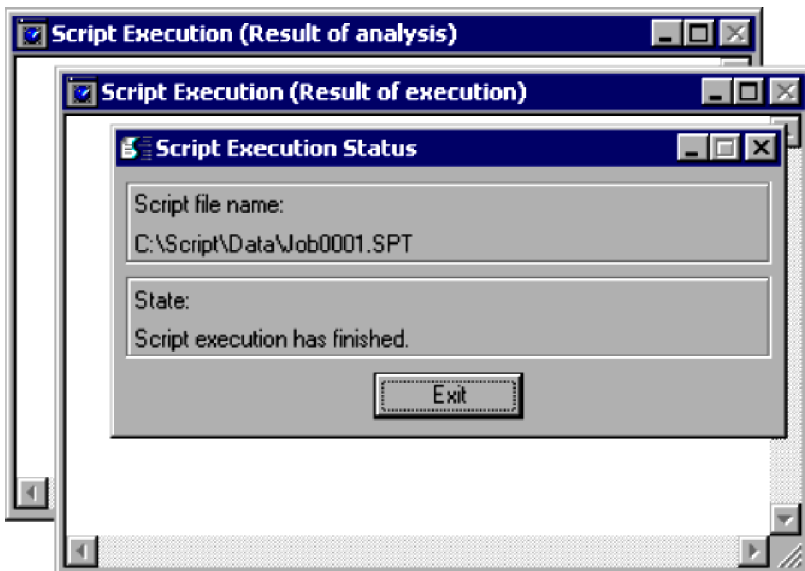
3. If you need to set the execution environment, click the **Detail** button.
The Set Execution Environment (Start Information) dialog box appears. Set the command line and other information.



For details about using this dialog box, see [4.1.8 Set Execution Environment \(Start Information\) dialog box](#).

4. In the Execution dialog box, click the **OK** button.

While the script is executing, windows show the execution status, execution trace, and analysis trace.



The script execution result is displayed as an execution trace.

When execution is completed, the **Exit** button is enabled in the window showing the execution status.

5. Check the contents of the execution trace.

6. Click the **Exit** button in the window displaying the execution status.

The three windows close.

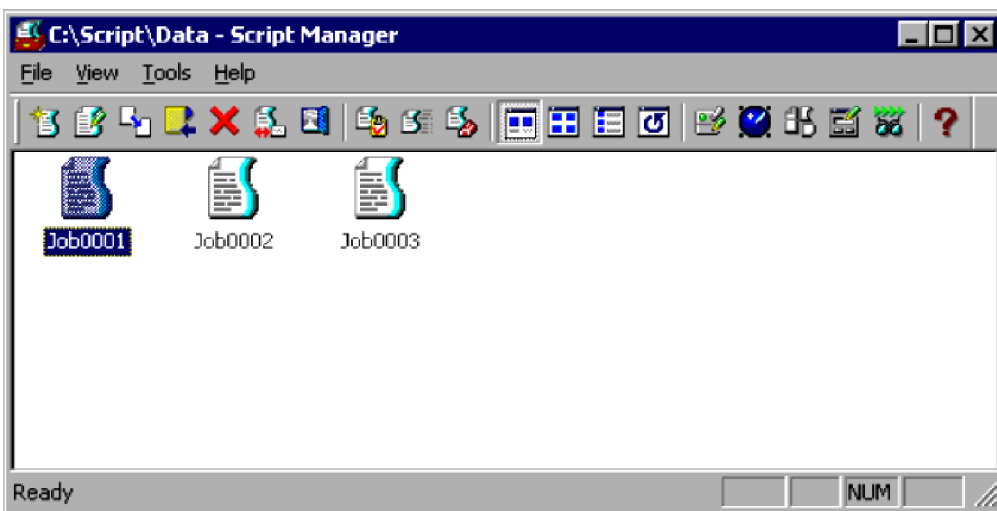
Notes

- When you execute a script in the Script Manager window, JP1/Script creates an execution trace file. To print this file, start Trace Viewer, select the trace file, and then choose **File, Print**. For details, see [3.4.10 Printing a trace file](#).
- If you click **Cancel** at step 4, the dialog box closes and the script is not executed.
- To view the three windows that show the execution status, execution trace, and analysis trace, select the **Display status window** check box. This check box is selected by default.

3.1.16 Creating a menu form

To create a menu form:

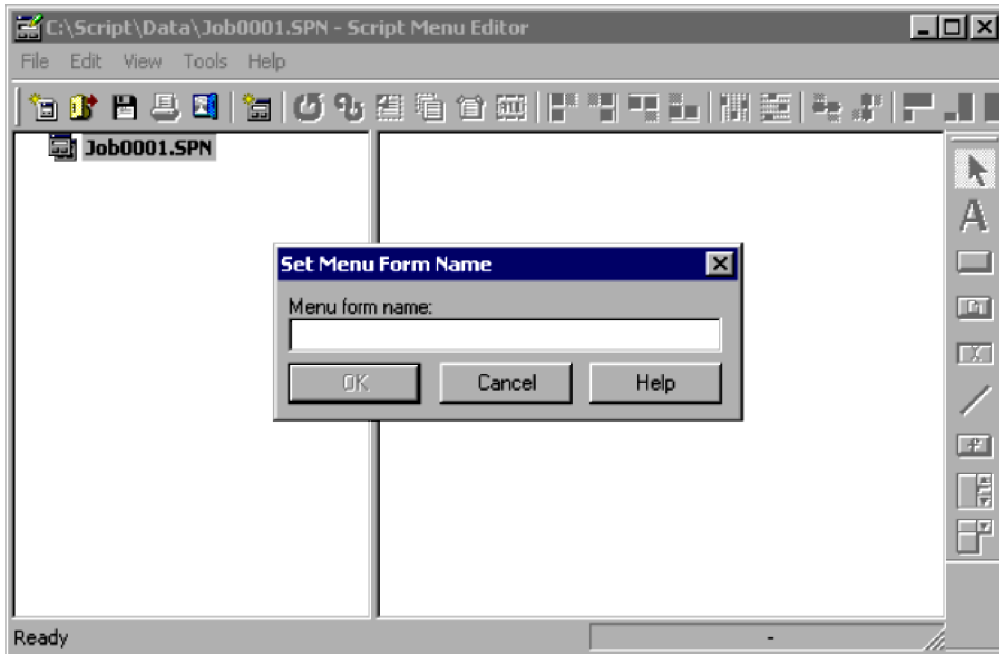
1. In the client area, select the icon of the script file you want to edit or execute.



2. Choose **File, Start Menu Editor**.

The Script Menu Editor window appears.

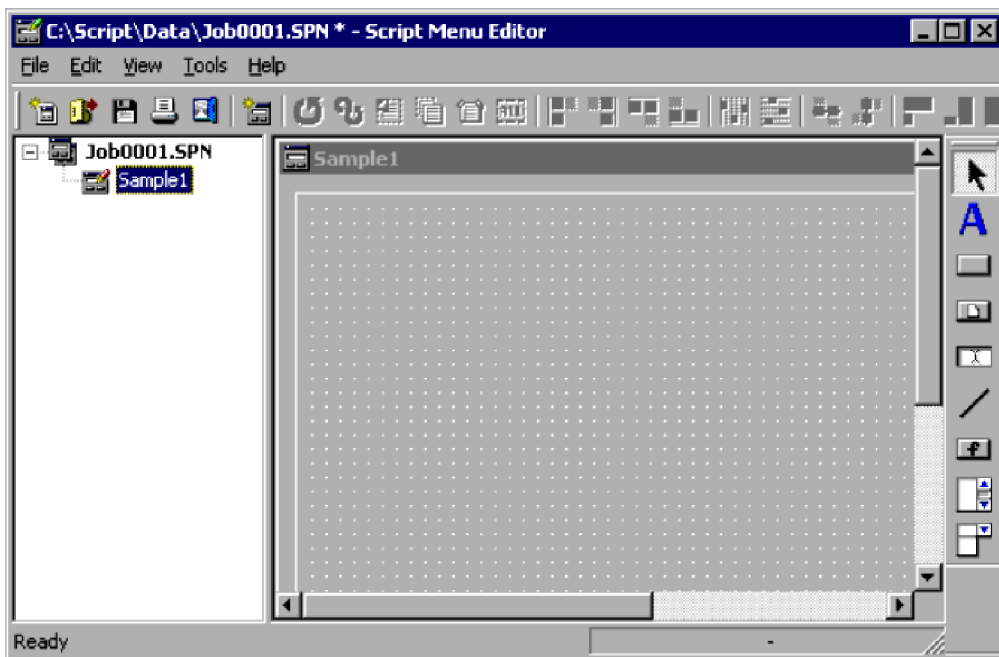
If no menu form has been created, the Set Menu Form Name dialog box appears in front of the Script Menu Editor window. The Command Properties dialog box appears at the bottom left of the window.



If a menu form has already been created, go to step 4.

3. Enter the menu form name.

A new menu form appears in the form view area. Proceed to step 5.



4. Choose **Edit, New Menu Form**.

A new menu form appears in the form view area.

5. Create the menu form.

For details on the procedure, see [3.6.3 Creating a menu form](#).

6. Quit Menu Editor.

Note

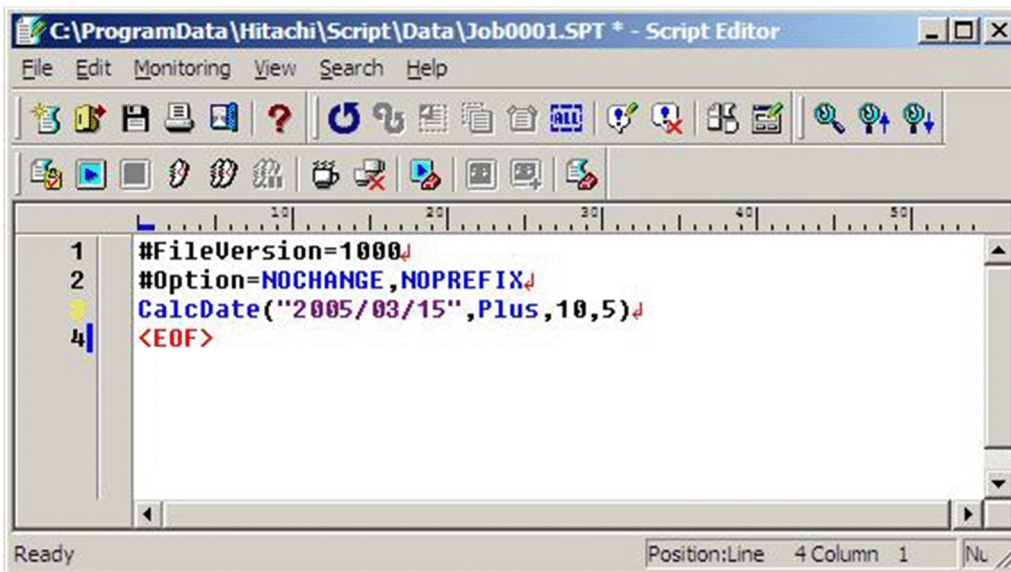
Manager and Menu Editor work independently. If you quit Manager after starting Menu Editor, Menu Editor does not terminate.

3.1.17 Starting and stopping Editor

- Starting Editor
Start Editor to actually begin creating or editing a script file.

To start Editor:

- In the client area, select the icon of a script file.
- In the Script Manager window, choose **File** and then **Edit**.



If a different editor is linked, that editor starts.

- Stopping Editor
In the Manager window, choose **File**, **Exit**. Script Editor stops.

3.2 Editor operations

You can perform the following operations in the Script Editor window:

- Set or cancel a comment line
- Use the Easy Input facility
- Create a Script menu form
- Set the Editor operating environment
- Check script syntax
- Start and cancel monitoring
- Execute a script
- Set and cancel breakpoints in monitoring mode
- Set the operating environment for monitoring mode
- Add a variable to the Watch window
- Set the script execution environment
- Search for text
- Replace text
- Create a new script file[#]
- Open an existing script file[#]
- Save a script file, replacing its previous contents[#]
- Save a script file under a new name[#]
- Print a script file[#]
- Undo the previous action[#]
- Redo the previous action[#]
- Cut selected text to the clipboard[#]
- Copy selected text to the clipboard[#]
- Paste text from the clipboard to a specified position[#]
- Select all written text[#]
- Jump to the line being executed[#]
- Show or hide the toolbar[#]
- Show or hide the status bar[#]
- Align the toolbars[#]
- Show or hide the ruler[#]
- Show or hide the vertical scroll bar[#]
- Show or hide line numbers[#]
- View the beginning of a file[#]

- View the end of a file#
- Show or hide the Watch window#
- Display online help#

#

These operations are not described. They are either standard Windows operations or simply involve choosing a command from a menu.

3.2.1 Script Editor window and menus

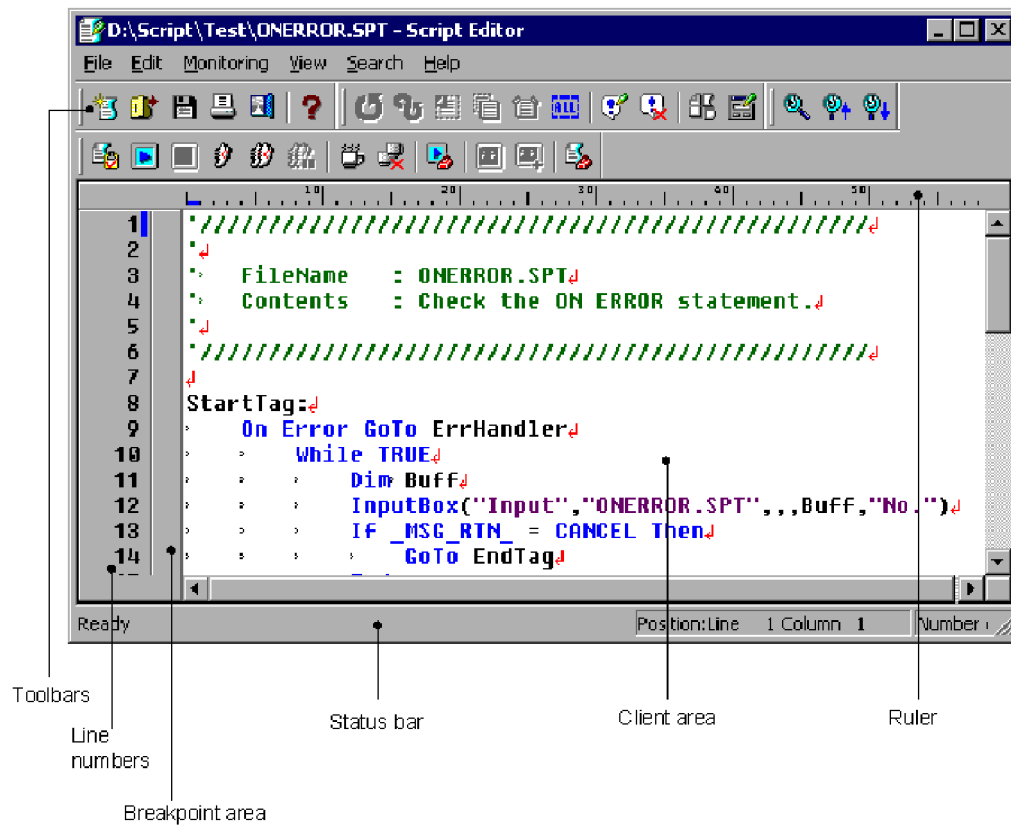
Editor is used for creating and editing scripts.

This section describes the functions and menus of the Script Editor window which appears when you start Editor.

(1) Script Editor window

Figure 3-2 shows the Script Editor window and the names of its components.

Figure 3–2: Script Editor window



(a) Toolbars

The toolbars contain buttons representing the most frequently used commands on the pull-down menus. By simply clicking a button, you can execute the corresponding command. You can hide a toolbar by toggling the corresponding **Toolbar** command in the **View** menu.

The following command buttons are displayed on the toolbars of the Script Editor window.

Standard toolbar

Create

Creates a new script file.

Open

Opens a script file.

Save

Saves a script file.

Print

Prints a script file.

Exit

Quits Script Editor.

Help

Displays the JP1/Script online help.

Editing toolbar

Undo

Reverses the previous action.

Redo

Reinstates the previous action.

Cut

Cuts selected text to the clipboard.

Copy

Copies selected text to the clipboard.

Paste

Pastes text from the clipboard to a specified position.

Select All

Selects all the written text.

Set Comment

Sets a selected line as a comment.

Cancel Comment

Changes a selected comment into a normal line.

Easy Input

Enables easy input of a command or statement.

Menu Editor

Starts Menu Editor.

Monitoring toolbar

Syntax Check

Checks the syntax of a completed script.

Execution Up to the Breakpoint

Starts or restarts execution up to the next breakpoint.

Cancel Monitoring

Cancels monitoring.

Step Execution

Executes commands or statements one by one.

Consecutive Step Execution

Executes a sequence of commands or statements.

Pause Consecutive Step Execution

Suspends consecutive step execution.

Set/Cancel Breakpoint

Sets or cancels a breakpoint.

Cancel All Breakpoints

Cancels all breakpoints.

Set

Sets the monitoring environment.

Show/Hide Watch Window

Shows or hides the Watch window.

Add to Watch Window

Adds a selected variable to the Watch window.

Set Execution Environment

Sets the script file environment.

Search toolbar**Find**

Sets the search criteria.

Find Previous

Searches for the string from the current position to the beginning of the script file.

Find Next

Searches from the current position to the end of the script file.

(b) Ruler

A scale showing the column numbers across the lines.

(c) Line numbers

An area showing the script line numbers.

(d) Breakpoint area

An area showing the breakpoint symbol (●) and the next-execution-step symbol (▶).

(e) Status bar

The status bar displays messages about the current processing being executed by Script Editor and status messages at completion of processing.

(f) Client area

The client area displays the script file contents.

(2) Menus in the Script Editor window

The pull-down and pop-up menus in the Script Editor window are described next.

(a) Menus and commands

Table 3-6 lists the Script Editor commands (functions) provided in the pull-down menus.

Table 3–6: Script Editor menus and commands

Menu	Command (function)	Description
File	Create	Creates a new script file.
	Open	Opens an existing script file.
	Save	Saves a script file, replacing its previous contents.
	Save As	Saves a script file under a new name.
	Print	Prints a file.
	Exit	Quits Script Editor.
	<i>file-name</i>	Displays the names of the most recently saved script files, to a maximum of nine file names.
Edit	Undo	Reverses the previous action.
	Redo	Reinstates the previous action.
	Cut	Cuts selected text to the clipboard.
	Copy	Copies selected text to the clipboard.
	Paste	Pastes text from the clipboard to a specified position.
	Select All	Selects all the written text.
	Set Comment	Sets a selected line as a comment.
	Cancel Comment	Changes a selected comment into a normal line.
	Easy Input	Enables easy input of a command or statement.
	Menu Editor	Starts Menu Editor.
	Options	Sets the Editor operating environment.
Monitoring	Syntax Check	Checks the syntax of a completed script.
	Execute Monitoring - Execution	In monitoring mode, starts or restarts script execution up to the next breakpoint.
	Execute Monitoring - Step Execution	In monitoring mode, executes commands or statements one by one.
	Execute Monitoring - Consecutive Step Execution	In monitoring mode, executes a sequence of commands or statements until the user chooses Pause Consecutive Step Execution .
	Execute Monitoring - Pause Consecutive Step Execution	Suspends consecutive step execution.
	Execution	Executes the script file to the end.

Menu	Command (function)	Description
Monitoring	Cancel Monitoring	Cancels monitoring execution.
	Set/Cancel Breakpoint	Sets or cancels a breakpoint.
	Cancel All Breakpoints	Cancels all breakpoints.
	Set	Sets the operating environment for monitoring mode.
	Add to Watch Window	Adds a selected variable to the Watch window.
	Set Execution Environment	Sets the script file environment.
	Jump to Execution Line	In monitoring mode, displays the line being executed.
View	Toolbar - Standard Toolbar	Shows or hides the standard toolbar.
	Toolbar - Edit Toolbar	Shows or hides the editing toolbar.
	Toolbar - Monitoring Toolbar	Shows or hides the monitoring toolbar.
	Toolbar - Search Toolbar	Shows or hides the search toolbar.
	Status Bar	Shows or hides the status bar.
	Align Toolbars	Arranges the toolbars in four rows.
	Ruler	Shows or hides the ruler.
	Vertical Scroll Bar	Shows or hides the vertical scroll bar.
	Horizontal Scroll Bar	Shows or hides the horizontal scroll bar.
	Display Line Numbers	Shows or hides line numbers.
	View Beginning of File	Displays the beginning of the script file.
	View End of File	Displays the end of the script file.
View Watch Window	Shows or hides the Watch window.	
Search	Find	Sets a search or replace string, and the search method.
	Replace	Replaces the search string with specified text.
	Find Previous	Searches for or replaces a string to the beginning of the file.
	Find Next	Searches for or replaces a string to the end of the file.
Help	Contents	Displays the contents of the JP1/Script online help.
	Search by Keyword	Lists the keywords of the JP1/Script online help.
	About JP1/Script	Displays version information for JP1/Script.

(b) Pop-up menu

To display the pop-up menu in the Script Editor window, right-click in the client area. Different pop-up menus are displayed in edit mode and monitoring mode. Table 3-7 lists the pop-up commands available in edit mode. Table 3-8 lists the pop-up commands in monitoring mode.

Table 3–7: Commands in the Script Editor pop-up menu (edit mode)

Command (function)	Description
Create	Creates a new script file.
Open	Opens an existing script file.

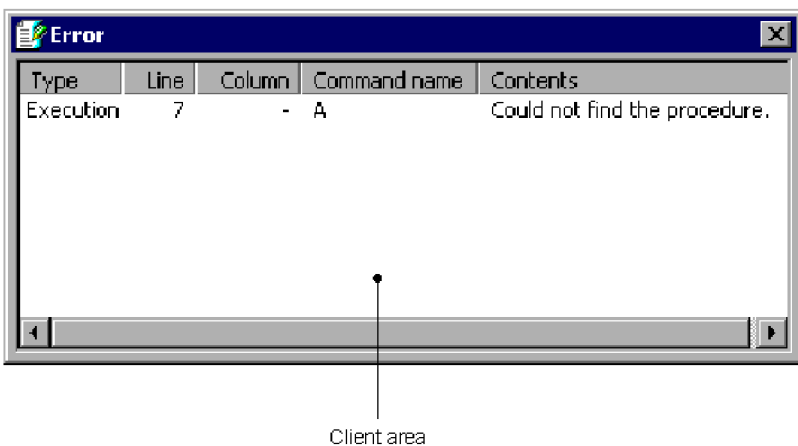
Command (function)	Description
Save	Saves a script file, replacing its previous contents.
Undo	Reverses the previous action.
Redo	Reinstates the previous action.
Cut	Cuts selected text to the clipboard.
Copy	Copies selected text to the clipboard.
Paste	Pastes text from the clipboard to a specified position.
Select All	Selects all the written text.
Set Comment	Sets a selected line as a comment.
Cancel Comment	Changes a selected comment into a normal line.
Easy Input	Enables easy input of a command or statement.
Menu Editor	Starts Menu Editor.

Table 3–8: Commands in the Script Editor pop-up menu (monitoring mode)

Command (function)	Description
Copy	Copies selected text to the clipboard.
Set/Cancel Breakpoint	Set or cancels a breakpoint.
Add to Watch Window	Adds a selected variable to the Watch window.

(3) Error window

The Error window displays errors occurring during monitoring execution.



(a) Client area

Type

Shows the type of error. When you double-click in this field, the cursor moves to the corresponding error.

Line

Shows the line in which the error occurred.

Column

Shows the column in which the error occurred.

Command name

Shows the command in which the error occurred.

Contents

Describes the error.

(b) Error window operations

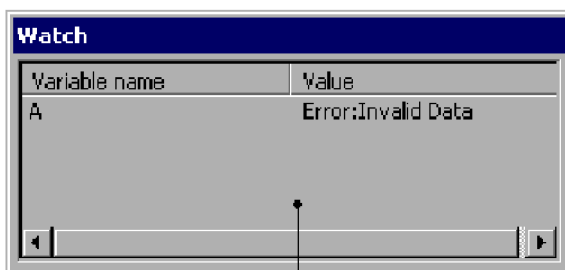
When you right-click in the Error window, the following pop-up menu appears:

- **Jump:** Jumps to the selected line or column.

(4) Watch window

Displays variable names and values when you are monitoring script execution.

For details about adding variables to the Watch window, see [3.2.12 Adding a variable to the Watch window](#).



(a) Client area

Variable name

Shows the variable name.

Value

Shows the variable value. If the variable name is invalid, `Error:Invalid Data` appears. If one of the following conditions exists for an array variable, `Error:Invalid Data` also appears because the value cannot be referenced:

- An index number is not specified.
- The specified index number is outside the valid range.
- The specified array variable does not have an appropriate dimension.

(b) Watch window operations

When you right-click in the Watch window, the following pop-up menu appears:

- **Update Value:** Updates the value of the selected variable.
- **Delete Variable:** Deletes the selected variable.

3.2.2 Mouse and key operations in the Script Editor window

(1) Mouse operations

Table 3-9 lists mouse operations in the client area of the Script Editor window.

Table 3–9: Mouse operations in the Script Editor window

Operation	Processing
Click	Clears the previous selection and makes a new selection.
Double-click	Selects text.
Right-click	Displays a pop-up menu.

(2) Key operations

Table 3-10 lists key operations in the client area of the Script Editor window and shows whether each operation is supported in edit mode and monitoring mode.

Table 3–10: Key operations in the Script Editor window

Operation	Edit mode	Monitoring mode	Processing
Ctrl + A	Y	Y	Selects all the written text.
Ctrl + C	Y	Y	Copies selected text.
Ctrl + E	Y	N	Starts Menu Editor.
Ctrl + F	Y	Partly	Sets a search or replace string.
Ctrl + H	Y	N	Replaces the search string with specified text.
Ctrl + K	Y	N	Enables easy input of a command or statement.
Ctrl + M	Y	N	Sets a selected line as a comment.
Ctrl + N	Y	N	Creates a new script file.
Ctrl + O	Y	N	Opens an existing script file.
Ctrl + P	Y	N	Prints a script file.
Ctrl + S	Y	N	Saves a script file.
Ctrl + T	Y	N	Sets the operating environment for monitoring mode.
Ctrl + V	Y	N	Pastes text from the clipboard.
Ctrl + X	Y	N	Cuts selected text.
Ctrl + Z	Y	N	Undoes the previous edit.
Ctrl + F5	Y	N	Executes a script file.
Ctrl + F11	Y	Y	Starts consecutive step execution.
Ctrl + Home	Y	Y	Displays the beginning of the script file.
Ctrl + End	Y	Y	Displays the end of the script file.
F1	Y	Y	Displays the Script help.#

Operation	Edit mode	Monitoring mode	Processing
F3	Y	Partly	Searches for or replaces a string to the end of the file.
F5	Y	Y	Starts or restarts execution up to the next breakpoint.
F7	Y	N	Performs a syntax check.
F9	Y	Y	Sets or cancels a breakpoint.
F11	Y	Y	Executes commands or statements one by one.
Alt + 0	N	Y	Adds a selected variable to the Watch window.
Alt + 1	N	Y	Shows or hides the Watch window.
Alt + F4	Y	Y	Quits Script Editor.
Shift + F3	Y	Partly	Searches for or replaces a string to the beginning of the file.
Shift + F5	N	Y	Cancels monitoring execution.
Shift + F9	Y	Y	Cancels all breakpoints.
Shift + Ctrl + M	Y	N	Changes a selected comment into a normal line.
Shift + Ctrl + Z	Y	N	Redoes the previous edit.
Shift + Ctrl + F11	N	Y	Suspends consecutive step execution.
Shift + Ctrl + Enter	Y	N	Sets the script file environment.

Y: Function can be performed.

Partly: Function can be partially performed.

N: Function cannot be performed.

#: When you press the F1 key with the cursor placed on a command or statement, help about that command or statement appears.

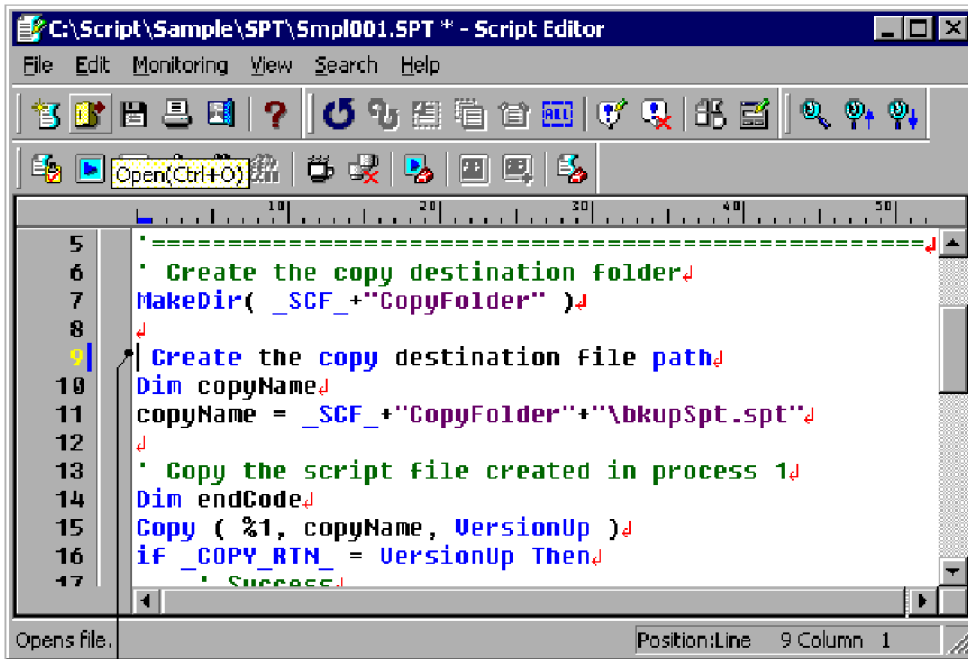
3.2.3 Setting and canceling comment lines

You can set or cancel a comment line.

- Setting a comment line

To set a comment line:

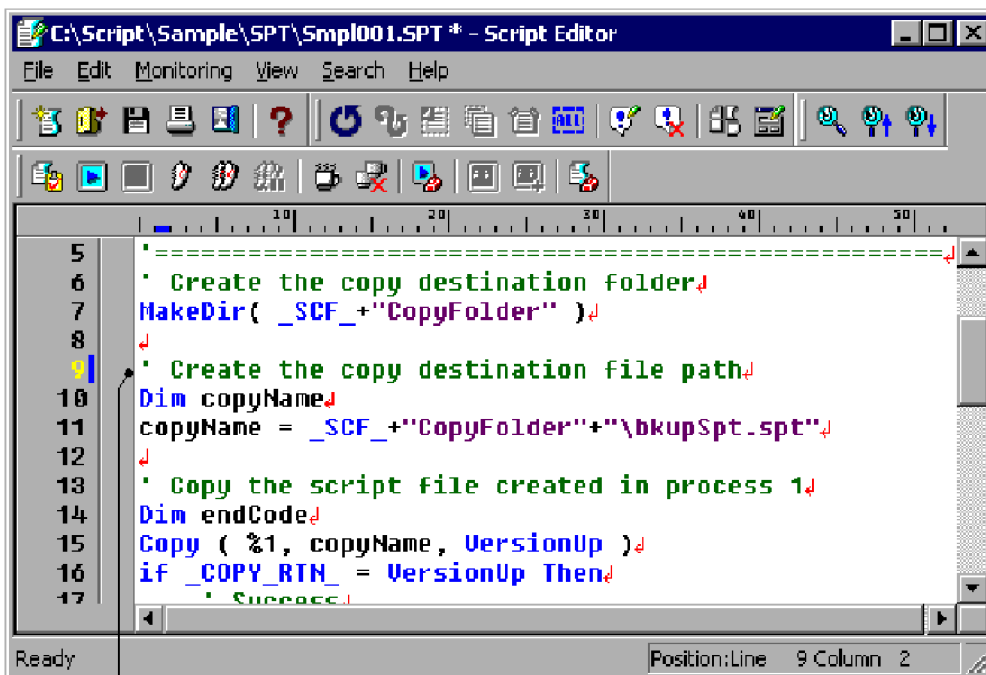
1. Move the cursor to the line you want to be a comment.



Cursor

2. Choose **Edit, Set Comment**.

The line where the cursor is located becomes a comment.



The apostrophe (') set at the beginning of the line indicates a comment.

- Canceling a comment line

To make a comment line into a normal line:

1. Move the cursor to the line from which you want to clear the comment setting.

The screenshot shows the Script Editor window titled "C:\Script\Sample\SPT\Smpl001.SPT - Script Editor". The menu bar includes File, Edit, Monitoring, View, Search, and Help. The toolbar contains various icons for file operations and execution. The script content is as follows:

```

5 |-----|
6 | ' Create the copy destination folder↓
7 | MakeDir( _SCF_"CopyFolder" )↓
8 | ↓
9 | ' Create the copy destination file path↓
10| Dim copyName↓
11| copyName = _SCF_"CopyFolder"+"bkupSpt.spt"↓
12| ↓
13| | Copy the script file created in process 1↓
14| Dim endCode↓
15| Copy ( %1, copyName, VersionUp )↓
16| if _COPY_RTN_ = VersionUp Then↓
17| | Success↓

```

The status bar at the bottom indicates "Ready" and "Position:Line 13 Column 1".

2. Choose **Edit, Cancel Comment**.

The comment setting is removed from the line where the cursor is located.

The screenshot shows the same Script Editor window, but the comment on line 13 has been removed. The script content is now:

```

5 |-----|
6 | ' Create the copy destination folder↓
7 | MakeDir( _SCF_"CopyFolder" )↓
8 | ↓
9 | ' Create the copy destination file path↓
10| Dim copyName↓
11| copyName = _SCF_"CopyFolder"+"bkupSpt.spt"↓
12| ↓
13| | Copy the script file created in process 1↓
14| Dim endCode↓
15| Copy ( %1, copyName, VersionUp )↓
16| if _COPY_RTN_ = VersionUp Then↓
17| | Success↓

```

The status bar at the bottom indicates "Ready" and "Position:Line 13 Column 1".

Notes

- If you select multiple lines, all the selected lines are set or canceled as comments.
- In monitoring mode, the **Set Comment** and **Cancel Comment** commands are disabled and you cannot choose them from the **Edit** menu.

3.2.4 Using Easy Input

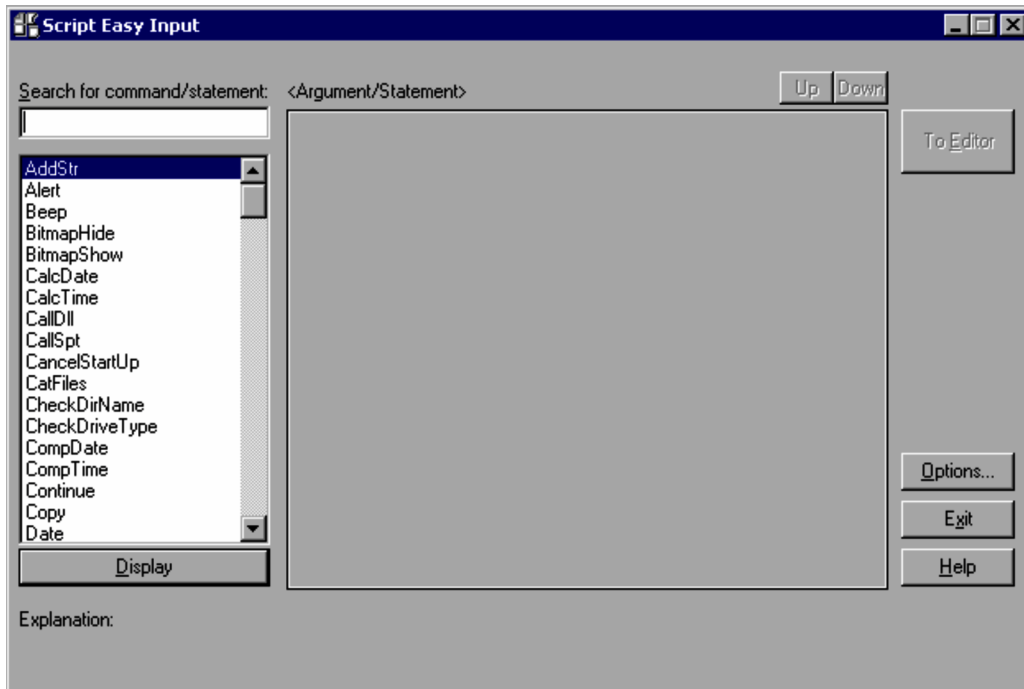
You can use the Easy Input program to enter commands and statements in a script file. With Easy Input, you can enter commands easily and accurately without needing any command-specific knowledge.

You can paste the entered commands and statements to an editor using the **To Editor** button.

To use Easy Input from Script Editor:

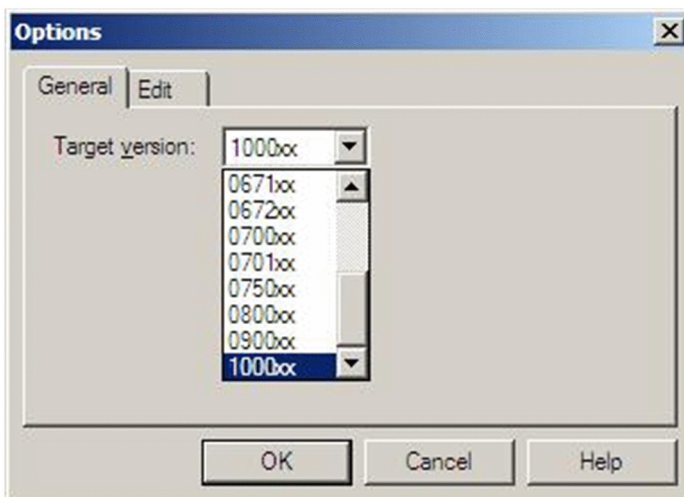
1. Choose **Edit, Easy Input** to start the Easy Input facility.

The Script Easy Input window appears.

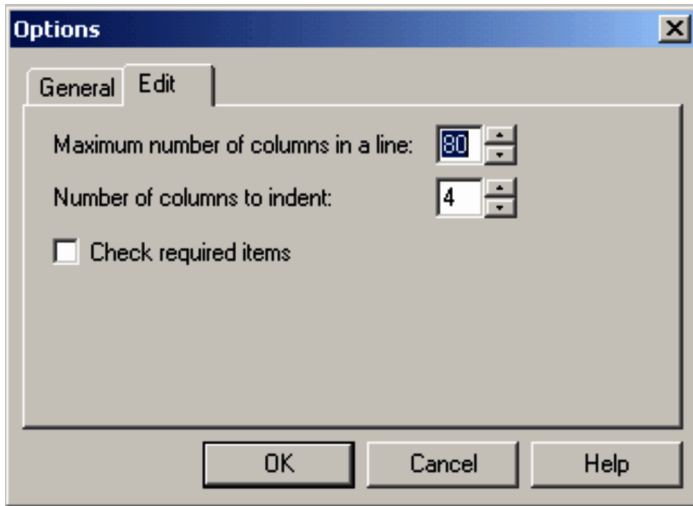


For details about using this window, see [3.3 Easy Input operations](#).

2. Click the **Options** button. In the **General** tab of the Options dialog box, set the version of the script file to be created.



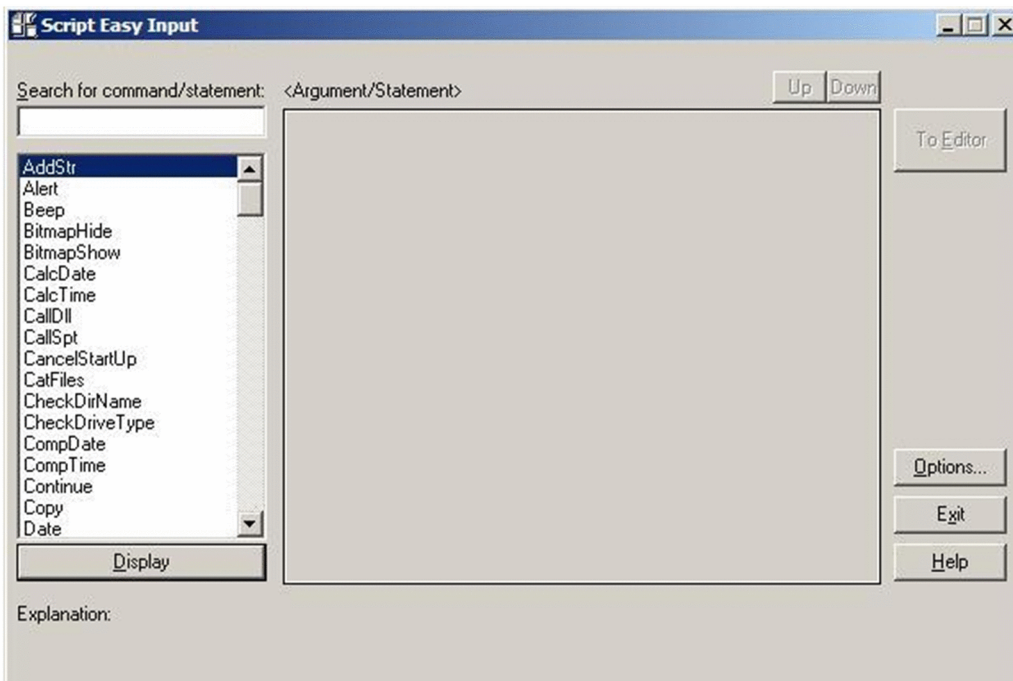
3. Click the **Edit** tab. In the **Edit** tab of the Options dialog box, set the format for output to Editor.



For details about how to set the dialog box, see [3.3.4 Setting the line length and indent size](#).

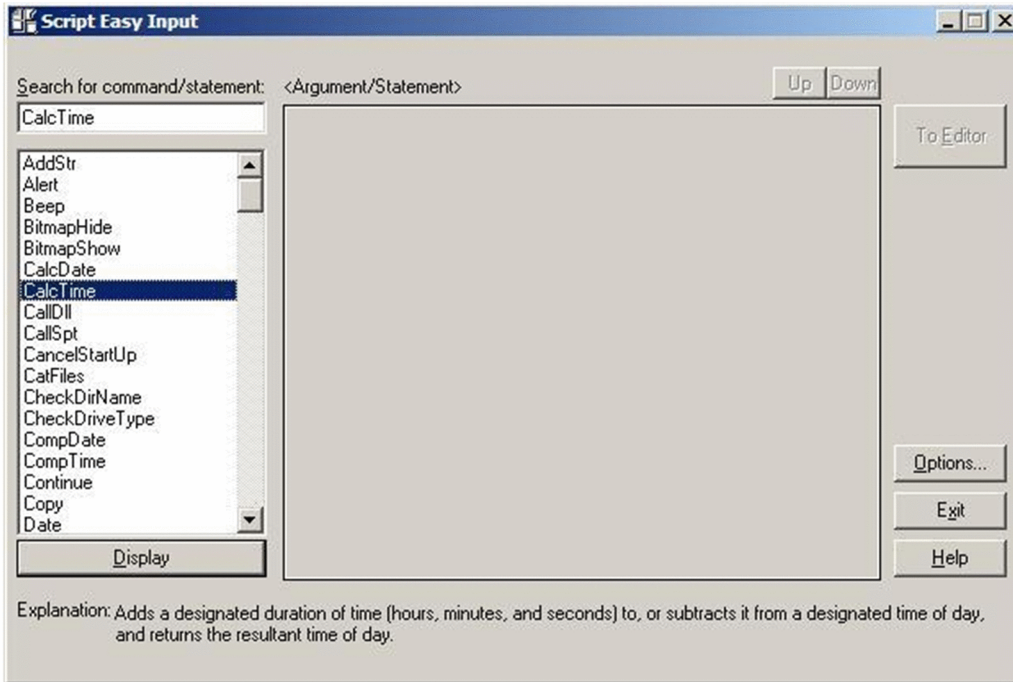
4. Click the **OK** button.

The script file version and clipboard output format are set. The Script Easy Input window appears again.



5. Find and select the command or statement you want to use.

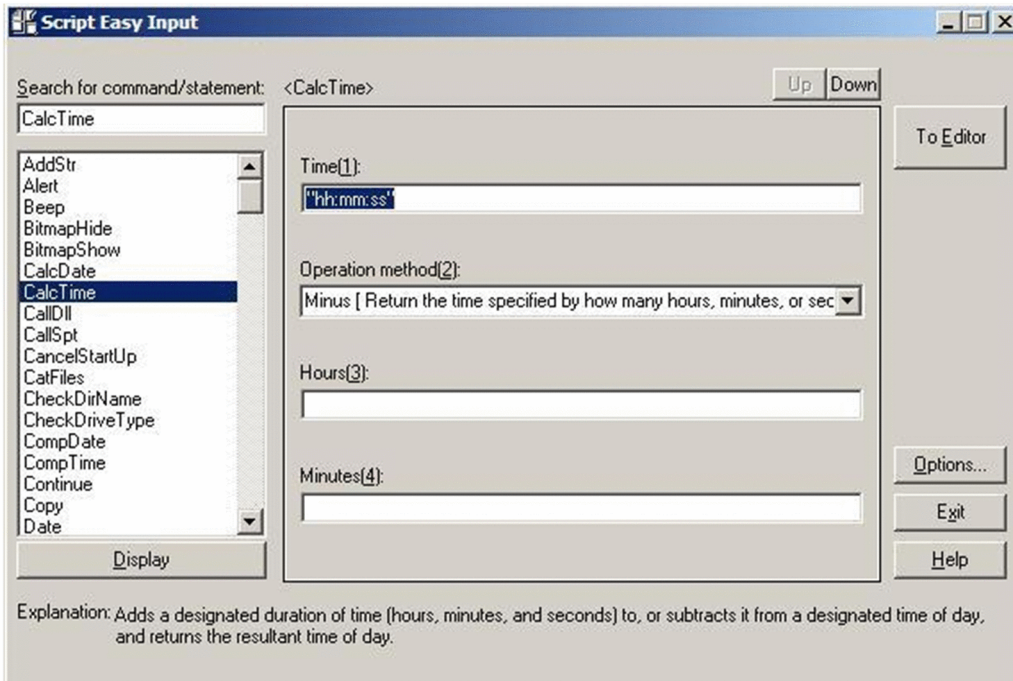
The focus shifts to the command or statement you selected. In this example, **CalcTime** is selected.



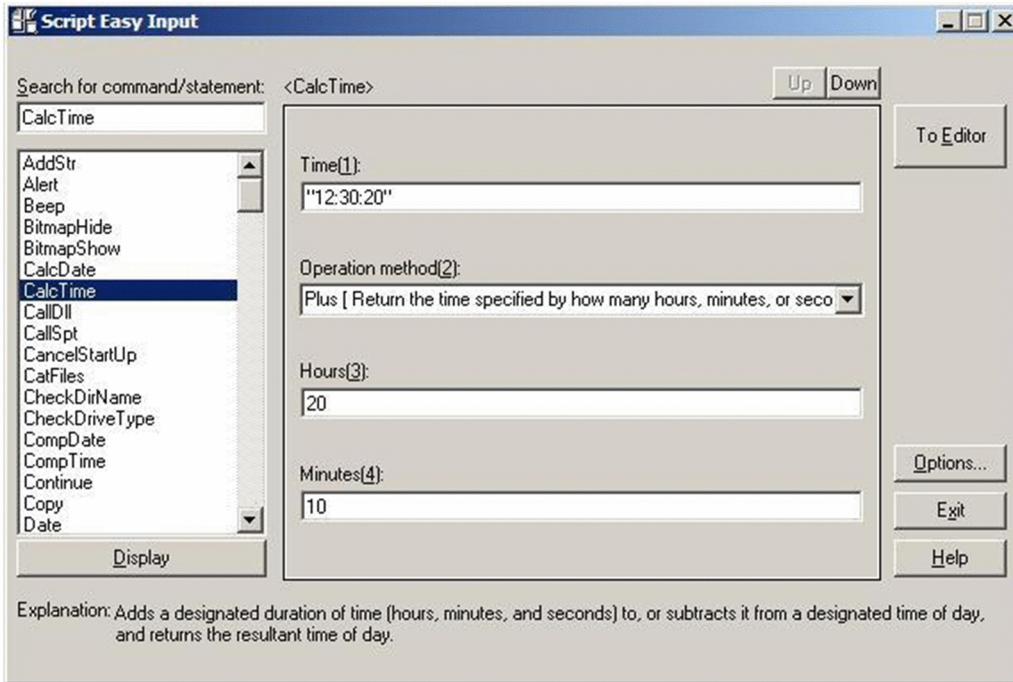
6. Click the **Display** button, or double-click the command or statement.

If you selected a command, the arguments that need to be set appear in the **Argument/Statement** area.

If you selected a statement, the statement syntax appears in this area.

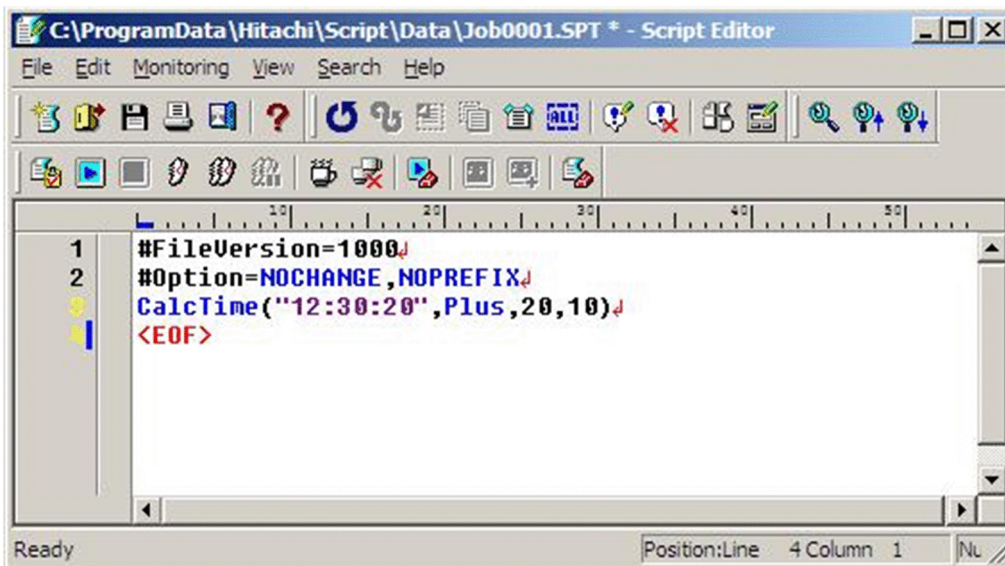


7. Enter the command arguments.



8. Click the **To Editor** button.

The contents that you set in the **Argument/Statement** area are pasted to Editor.



Notes

- Select a command name or statement name in the **Search for command/statement** list box and then right-click to view help about that command or statement.
- If the command you selected at step 5 has no arguments, a message to that effect appears in the **Argument/Statement** area.

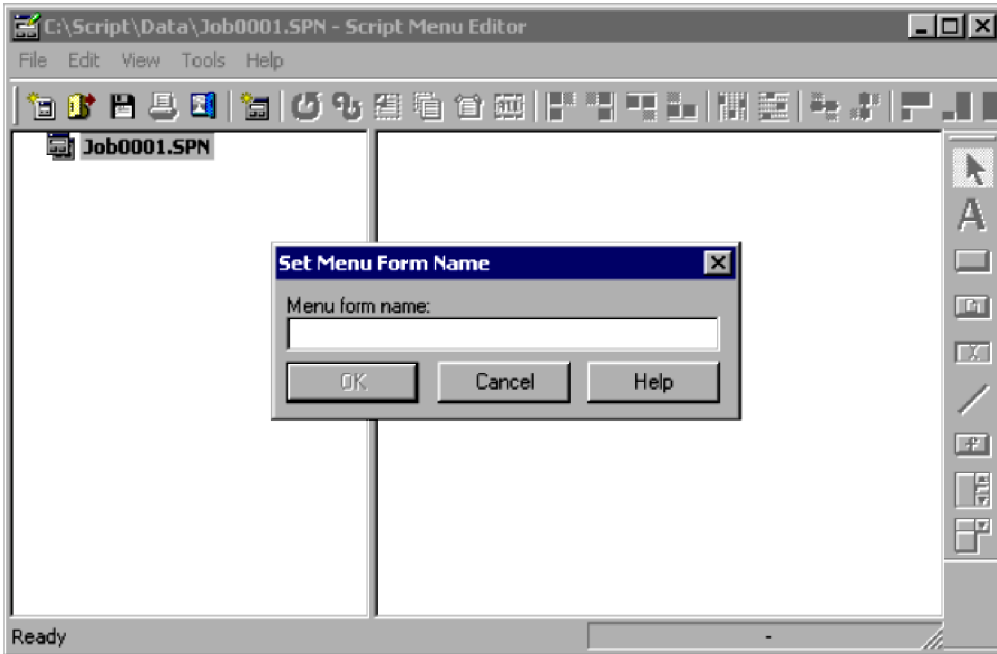
3.2.5 Creating a menu form

To create a menu form:

1. Choose **Edit, Menu Editor**.

The Script Menu Editor window appears.

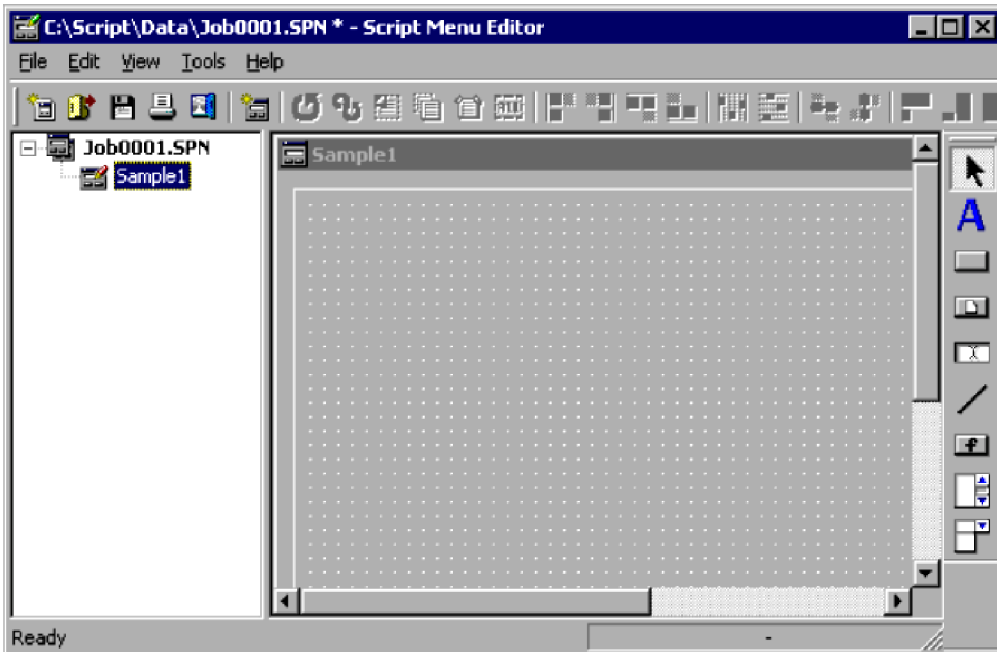
If no menu form has been created, the Set Menu Form Name dialog box appears in front of the Script Menu Editor window. The Command Properties dialog box appears at the bottom left of the window.



If a menu form has already been created, go to step 3.

2. Enter the menu form name.

A new menu form appears in the form view area. Proceed to step 4.



3. Choose **Edit, New Menu Form**.

A new menu form appears in the form view area.

4. Create the menu form.

For details on the procedure, see [3.6.3 Creating a menu form](#).

5. Quit Menu Editor.

Note

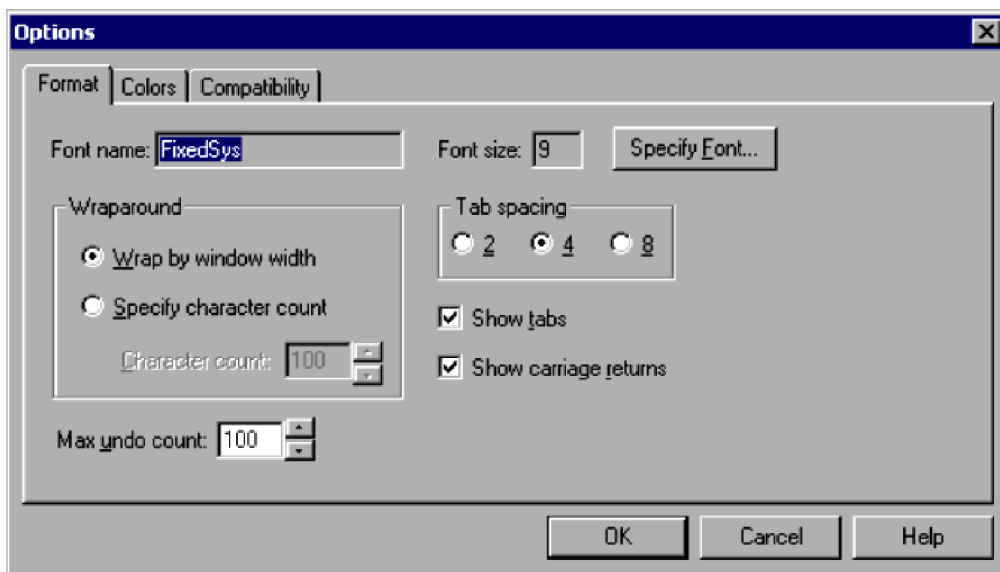
The **Edit, Menu Editor** command is disabled when Editor is in monitoring mode.

3.2.6 Setting the Editor operating environment

To set the Editor operating environment:

1. Choose **Edit, Options**.

The Options (Format) dialog box appears.

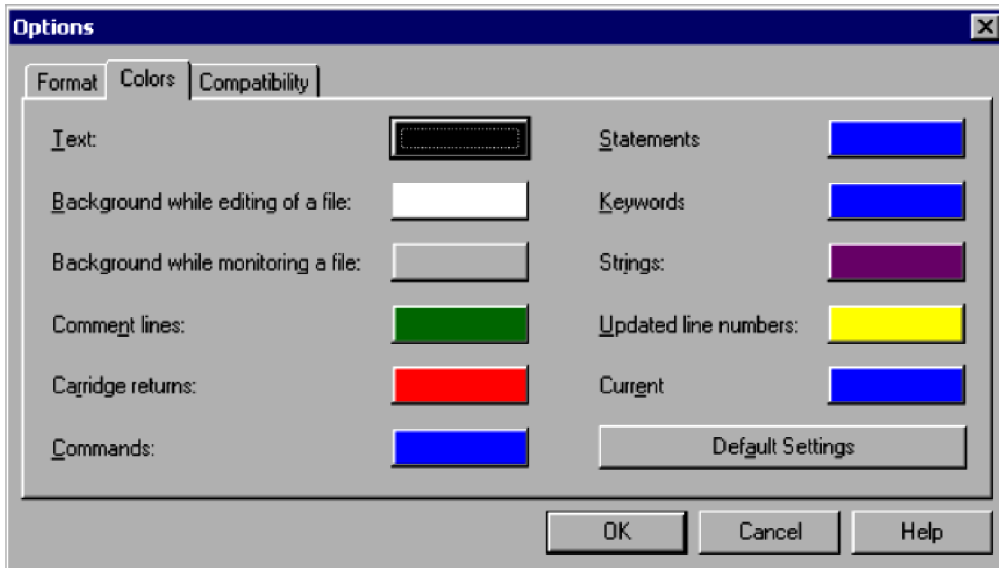


For details about using this dialog box, see [4.2.2 Options \(Format\) dialog box](#).

2. Set format information.

3. Click the **Colors** tab.

The Options (Colors) dialog box appears.



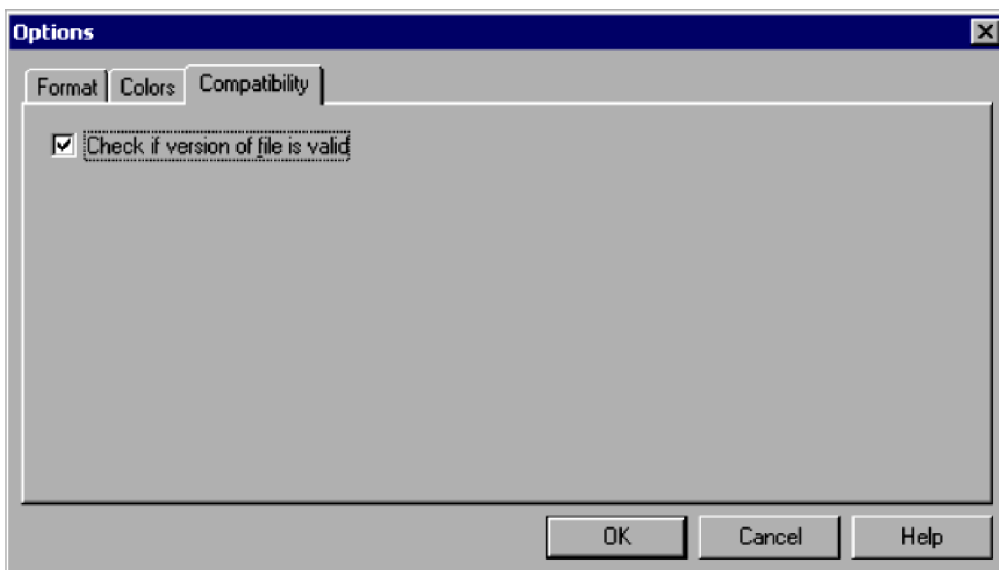
4. Set display color information.

To reset the display color of all items to the defaults, click **Default Settings**.

For details about using this dialog box, see [4.2.3 Options \(Colors\) dialog box](#).

5. Click the **Compatibility** tab.

The Options (Compatibility) dialog box appears.



6. Specify whether to check if the file version is specified at the head of the script file.

For details about using this dialog box, see [4.2.4 Options \(Compatibility\) dialog box](#).

7. Click the **OK** button.

The operating environment is set for Editor and the dialog box closes.

3.2.7 Checking script syntax

To check the syntax of a completed script file:

1. Choose **Monitoring, Syntax Check**.

Editor is set to monitoring mode and the syntax check starts.

The screen darkens momentarily.

```
12
13 ' Copy the script file created in process 1
14 Dim endCode
15 Copy ( %1, copyName, VersionUp )
16 if _COPY_RTN_ = VersionUp Then
17     ' Success
18     endCode = 0
19 elseif _COPY_RTN_ = Skip Then
20     'Skipped because the date is the same
21     endCode = 1
22 else
23     endCode = 2
```

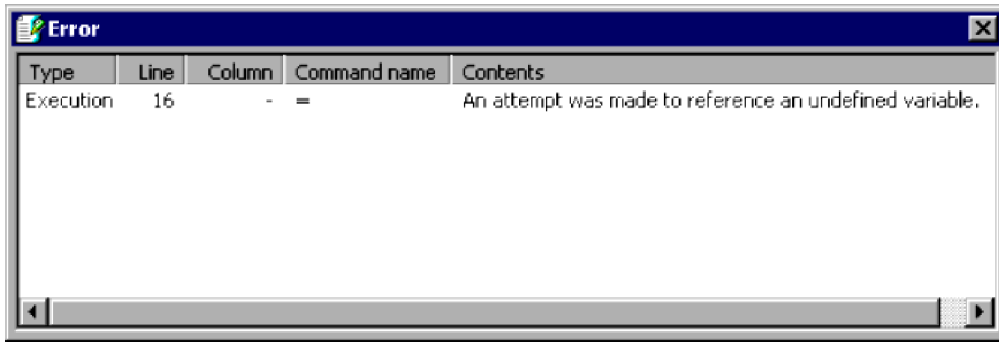
(Window display while the syntax check is in progress)

```
12
13 ' Copy the script file created in process 1
14 Dim endCode
15 Copy ( %1, copyName, VersionUp )
16 if _COPY_RTN_ = VersionUp Then
17     ' Success
18     endCode = 0
19 elseif _COPY_RTN_ = Skip Then
20     'Skipped because the date is the same
21     endCode = 1
22 else
23     endCode = 2
```

(Window display at syntax check completion)

Any syntax errors that are found appear in an Error window.

2. Check the contents of the Error window.



Notes

- The **Syntax Check** command is disabled when Script Editor is in monitoring mode.
- If you attempt to check the syntax of an unnamed script file, the Save As dialog box appears. You must save the file before you can perform the syntax check.
- If you have edited the script file, the syntax check is performed on the updated contents, but the updates are not saved.

3.2.8 Starting and canceling monitoring

Monitoring means tracking the operation of a script file while it is executing.

JP1/Script provides three monitoring methods.

Command	Description
Execution	Starts or restarts execution up to the next breakpoint.
Step Execution	Executes commands or statements one by one.
Consecutive Step Execution	Executes a sequence of commands or statements until the user chooses Pause Consecutive Step Execution .

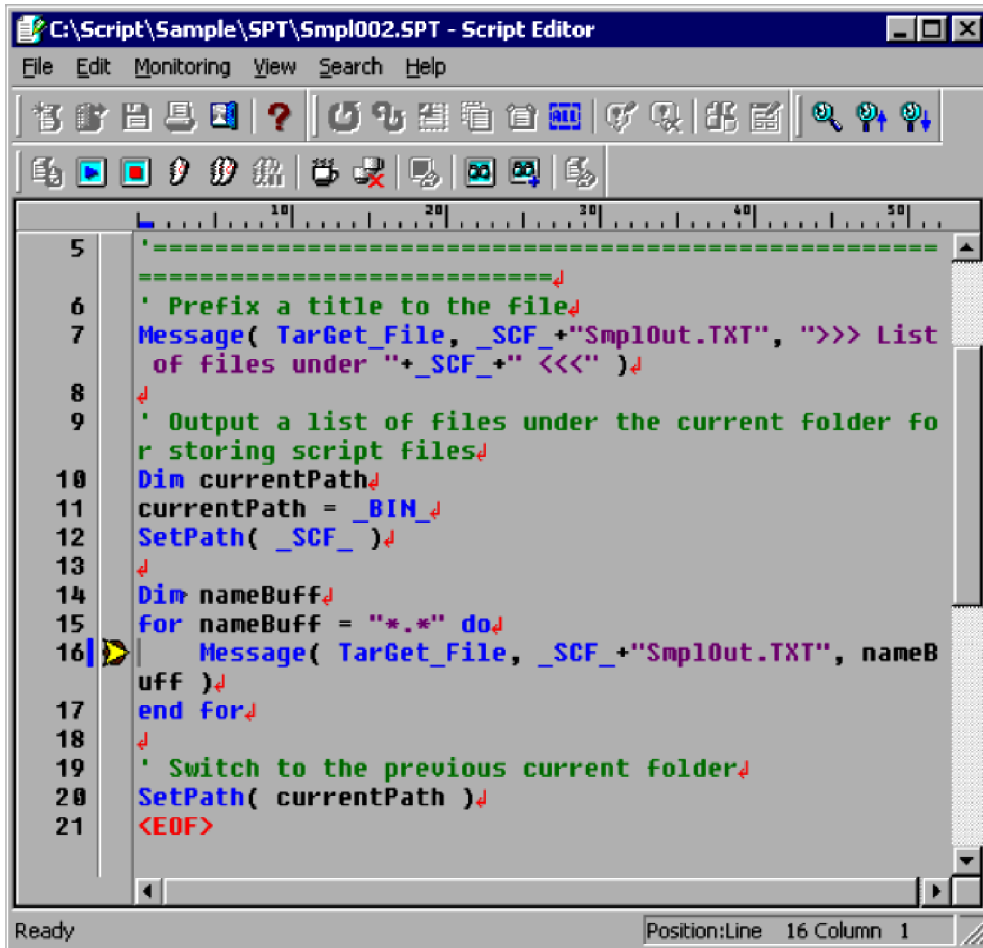
- Monitoring using the Execution method

To use the Execution monitoring method:

1. Choose **Monitoring**, **Execute Monitoring**, **Execution** or click the **Execution** button on the toolbar.

Script Editor enters monitoring mode and monitoring begins. An arrow symbol (➤) appears to the left of the next line to be executed. Comment lines and line spaces are ignored.

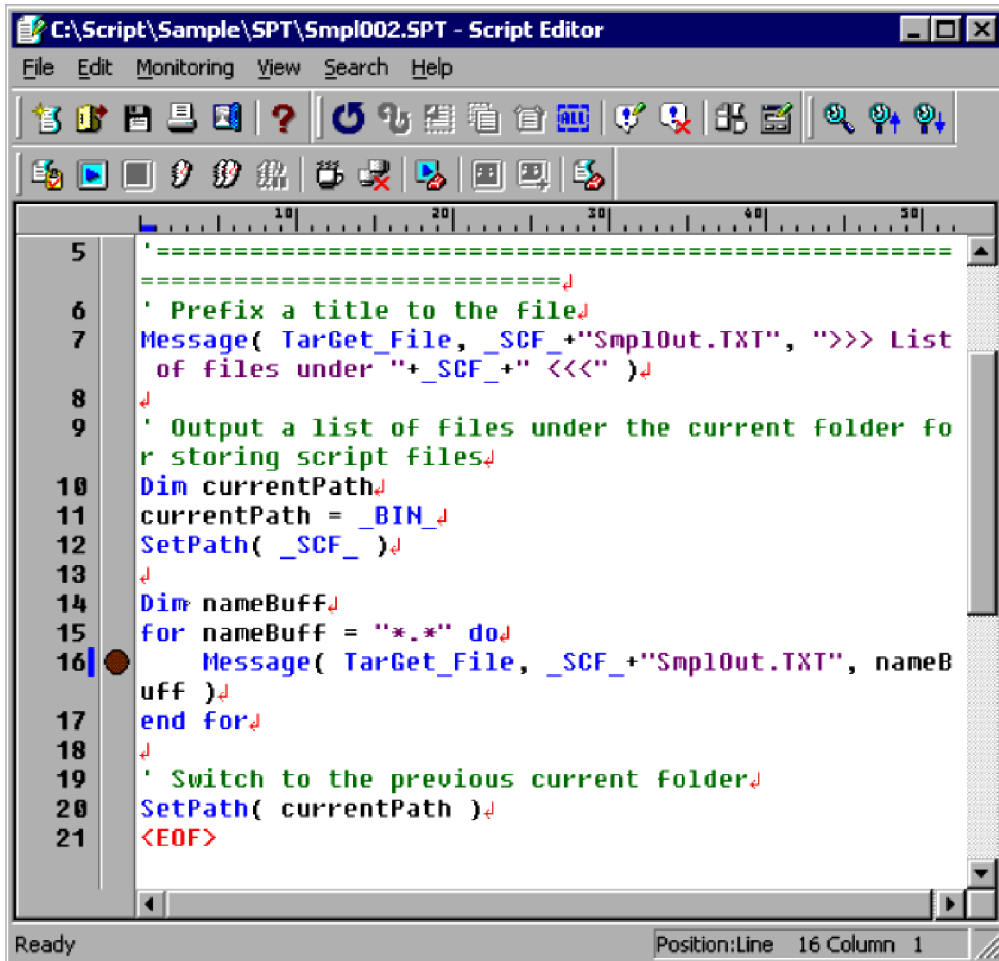
Execution continues until there is a breakpoint, indicated by a circle symbol (●). Execution is suspended at this point.



For details about using this dialog box, see [3.2.10 Setting and canceling breakpoints in monitoring mode](#).

2. To cancel monitoring, choose **Monitoring, Cancel Monitoring** or click the **Cancel Monitoring** button on the toolbar.

Monitoring immediately stops and Editor returns to edit mode.



- Monitoring using the Step Execution method

To use the Step Execution monitoring method:

1. Choose **Monitoring**, **Execute Monitoring**, **Step Execution** or click the **Step Execution** button on the toolbar. Editor enters monitoring mode and monitoring begins. An arrow symbol (➤) appears to the left of the next line to be executed. Comment lines and line spaces are ignored.

```
C:\Script\Sample\SPT\Smpl001.SPT - Script Editor
File Edit Monitoring View Search Help
[Toolbar icons]
=====
6 ' Create the copy destination folder
7 MakeDir( _SCF_+"CopyFolder" )
8
9 ' Create the copy destination file path
10 Dim copyName
11 copyName = _SCF_+"CopyFolder"+"bkupSpt.spt"
12
13 ' Copy the script file created in process 1
14 Dim endCode
15 Copy ( %1, copyName, VersionUp )
16 if _COPY_RTN_ = VersionUp Then
17     ' Success
18     endCode = 0
19 elseif _COPY_RTN_ = Skip Then
20     'Skipped because the date is the same
21     endCode = 1
22 else
23     endCode = 2
24 end
25
26 'Message ( Target_File, _SCF_+"Smpl001.txt", "Exit c
ode = "+endCode )
Ready Position:Line 10 Column 1
```

2. To cancel monitoring, choose **Monitoring, Cancel Monitoring** or click the **Cancel Monitoring** button on the toolbar.

Monitoring immediately stops and Editor returns to edit mode.

```

C:\Script\Sample\SPT\Smp1001.SPT - Script Editor
File Edit Monitoring View Search Help

=====↓
6 ' Create the copy destination folder↓
7 MakeDir( _SCF_+"CopyFolder" )↓
8 ↓
9 ' Create the copy destination file path↓
10 Dim copyName↓
11 copyName = _SCF_+"CopyFolder"+"bkupSpt.spt"↓
12 ↓
13 ' Copy the script file created in process 1↓
14 Dim endCode↓
15 Copy ( %1, copyName, VersionUp )↓
16 if _COPY_RTN_ = VersionUp Then↓
17     ' Success↓
18     endCode = 0↓
19 elseif _COPY_RTN_ = Skip Then↓
20     'Skipped because the date is the same↓
21     endCode = 1↓
22 else↓
23     endCode = 2↓
24 end↓
25 ↓
26 'Message ( Target_File, _SCF_+"Smp1001.txt", "Exit c
ode = "+endCode )↓

Ready Position:Line 10 Column 1

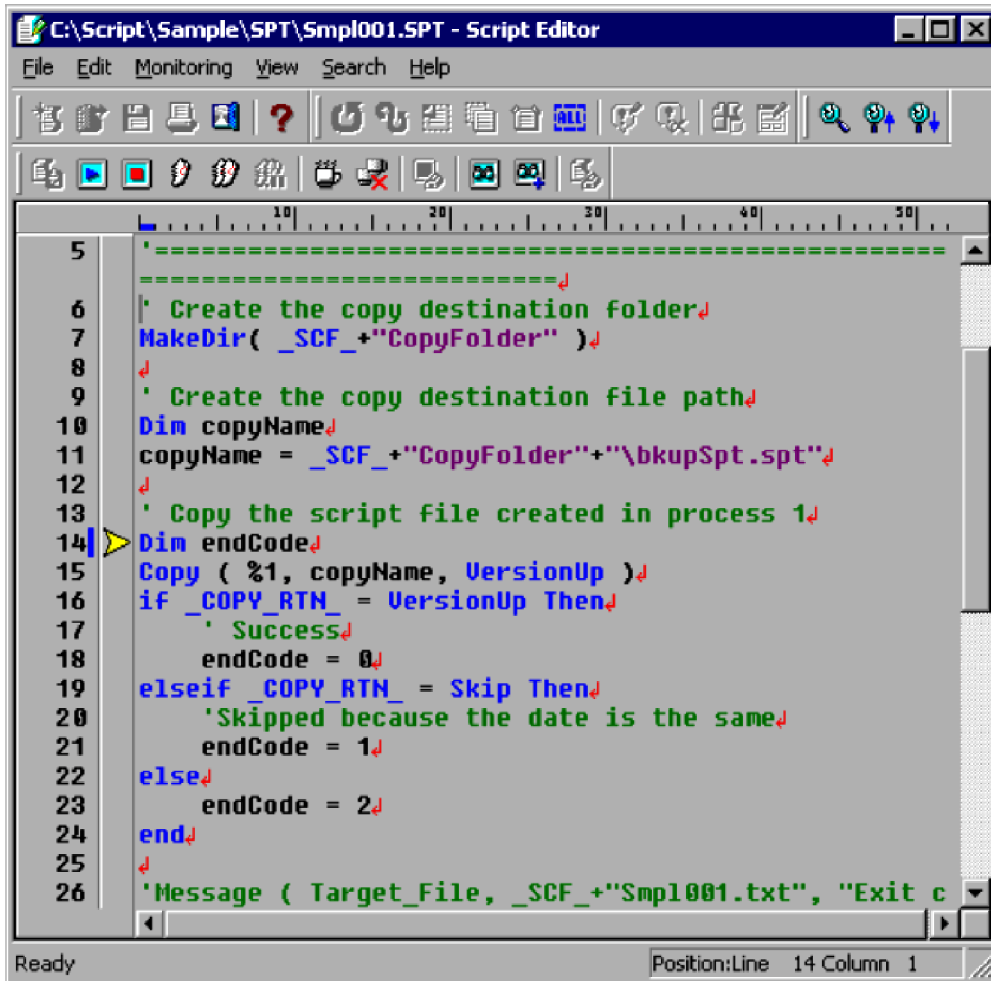
```

- Monitoring using the Consecutive Step Execution method

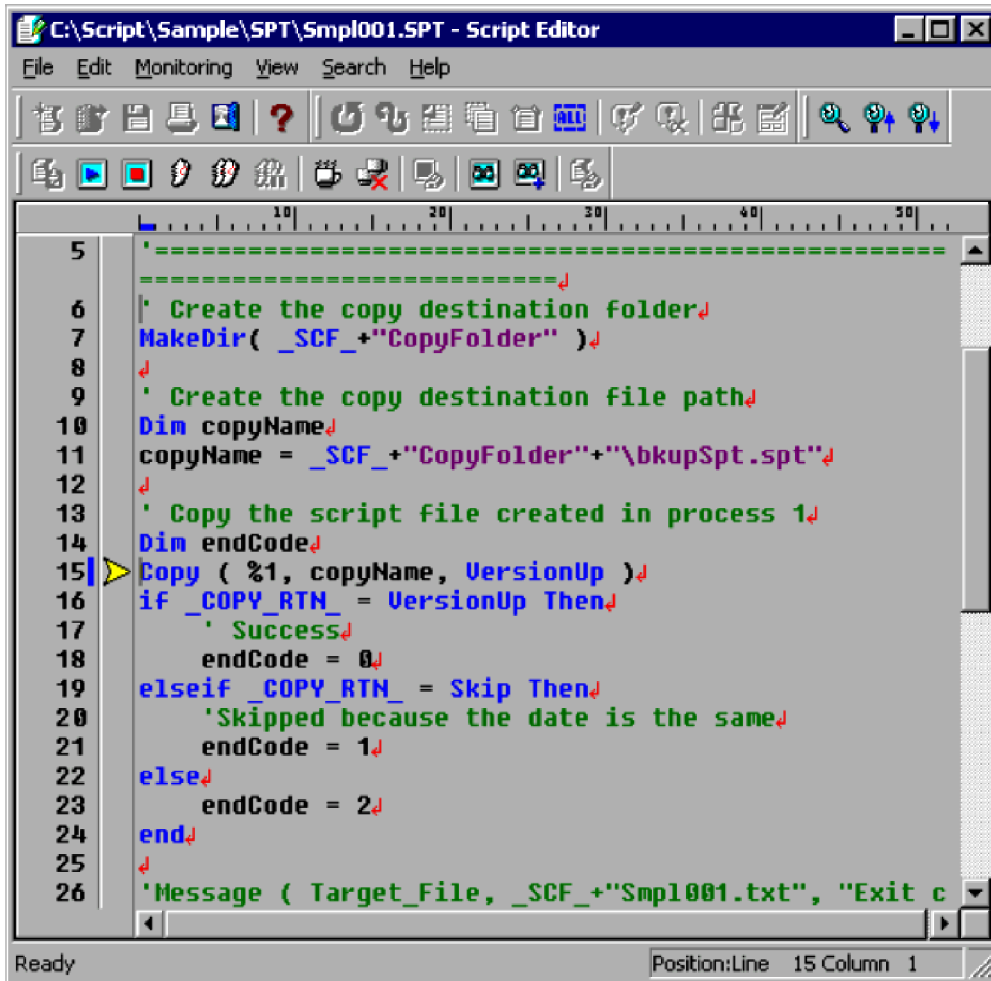
To use the Consecutive Step Execution monitoring method:

1. Choose **Monitoring**, **Execute Monitoring**, **Consecutive Step Execution** or click the **Consecutive Step Execution** button on the toolbar.

Editor enters monitoring mode and monitoring begins. An arrow symbol (➤) appears to the left of the next line to be executed. Comment lines and line spaces are ignored.

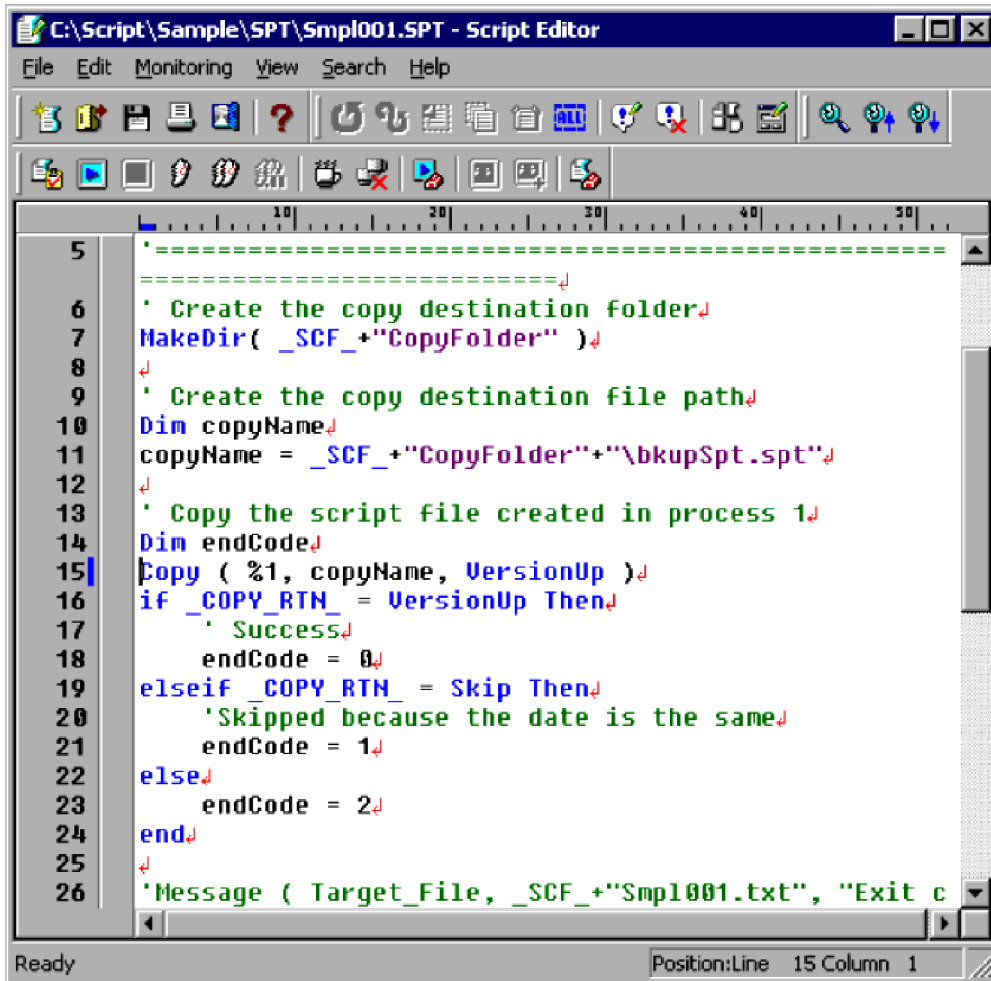


- To halt monitoring temporarily, choose **Monitoring, Execute Monitoring, Pause Consecutive Step Execution**. Alternatively, click the **Pause Consecutive Step Execution** button on the toolbar. Monitoring is temporarily suspended.



3. To cancel monitoring, choose **Monitoring, Cancel Monitoring** or click the **Cancel Monitoring** button on the toolbar.

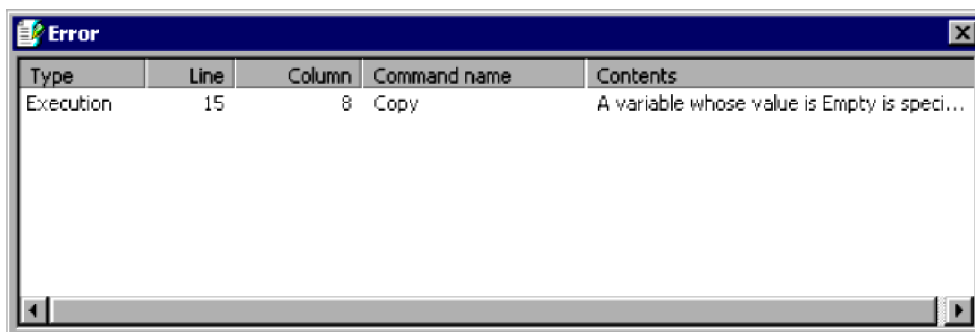
Monitoring immediately stops and Editor returns to edit mode.



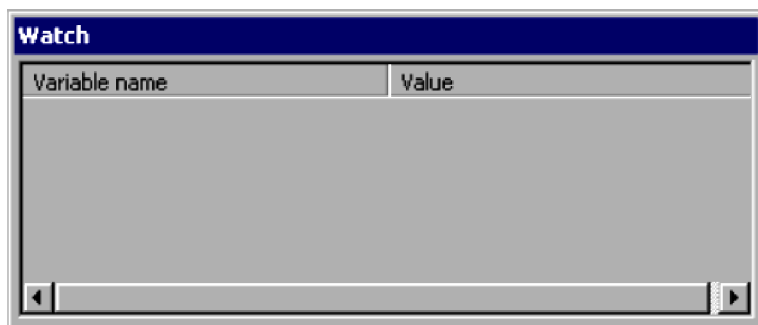
Notes

- If the script file enters an infinite loop, consecutive step execution will not stop when you choose **Monitoring, Execute Monitoring**, and then **Pause Consecutive Step Execution**. Instead, choose **Monitoring, Cancel Monitoring** to cancel execution.
- Syntax or execution errors found during monitoring are displayed in an Error window. Variable values added to the Watch window are successively updated and displayed during execution.

Error window example



- To view the Watch window, choose **View, View Watch Window** while monitoring a script. This command is available only during monitoring.
- Watch window example



- If you attempt to monitor an unnamed script file, the Save As dialog box appears. You must save the file before you can monitor its execution.
- If you have edited the script file, the syntax check is performed on the updated contents, but the updates are not saved.
- Do not monitor a script file for which the `On Error Goto` statement is specified.

3.2.9 Executing a script

The following describes script execution to the end of the file. This is ordinary execution, not monitoring.

To execute a script from the Script Editor window:

1. Choose **Monitoring, Execution**.

Script execution begins. Because monitoring is not being performed, no arrow symbol (➤) appears at the left of the next line to be executed.

Execution completes when the last line of the script has been executed.

Note

Because monitoring is not being performed, you cannot display the Watch window during execution by choosing **View, View Watch Window**.

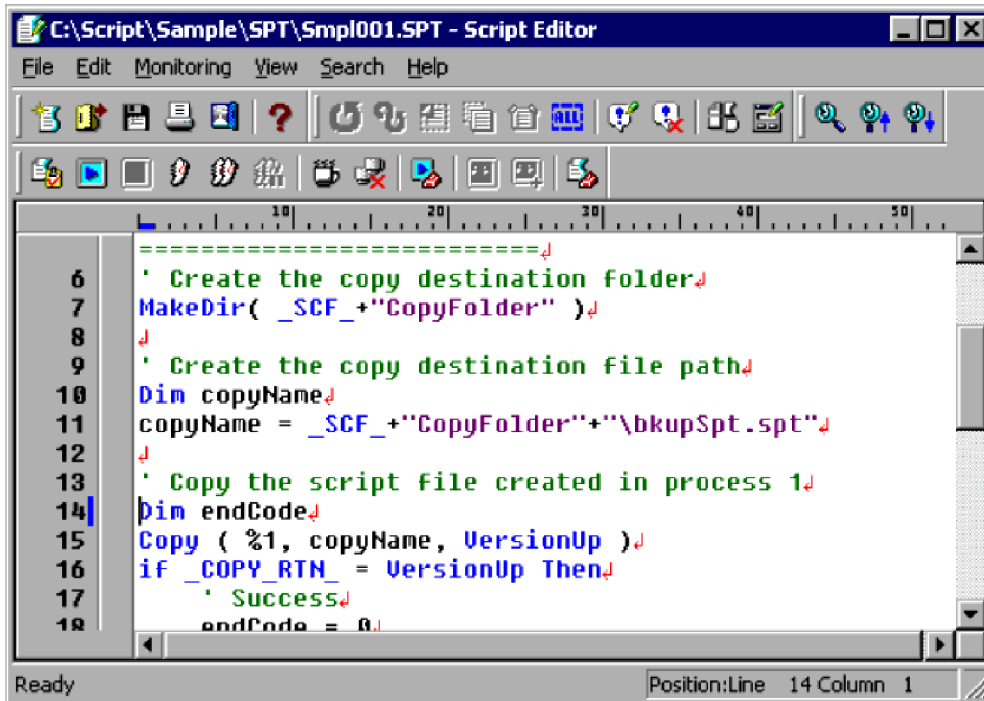
3.2.10 Setting and canceling breakpoints in monitoring mode

You can set a point at which execution in monitoring mode is to be temporarily suspended. You can also cancel a breakpoint.

- Setting a breakpoint

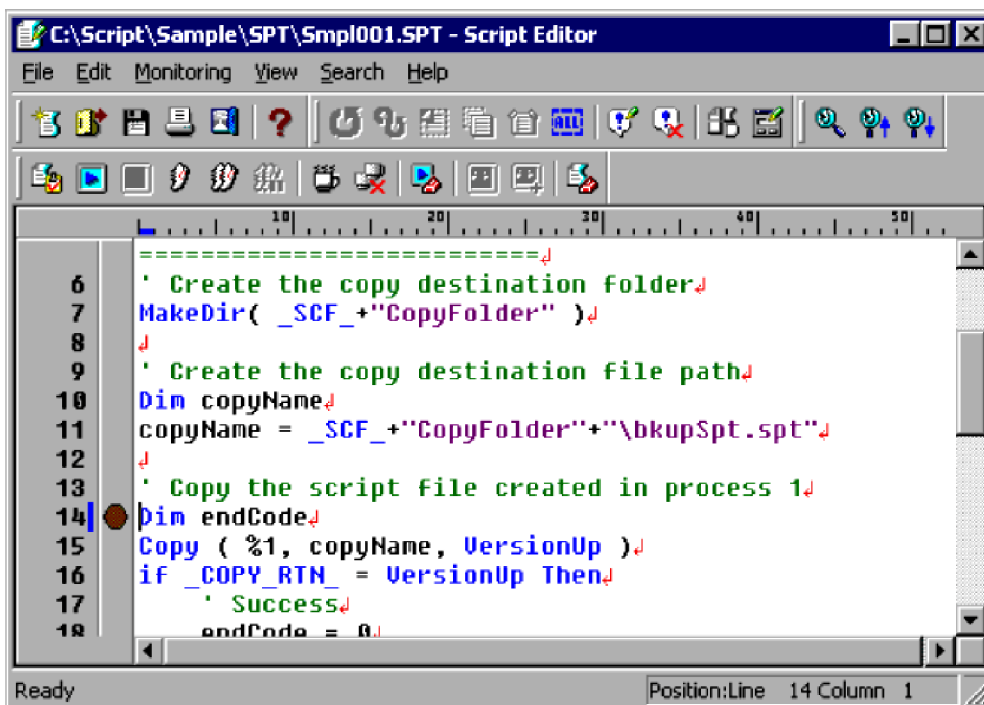
To set a breakpoint:

1. Move the cursor to the line at which you want to set a breakpoint.



2. Choose **Monitoring, Set Breakpoint**.

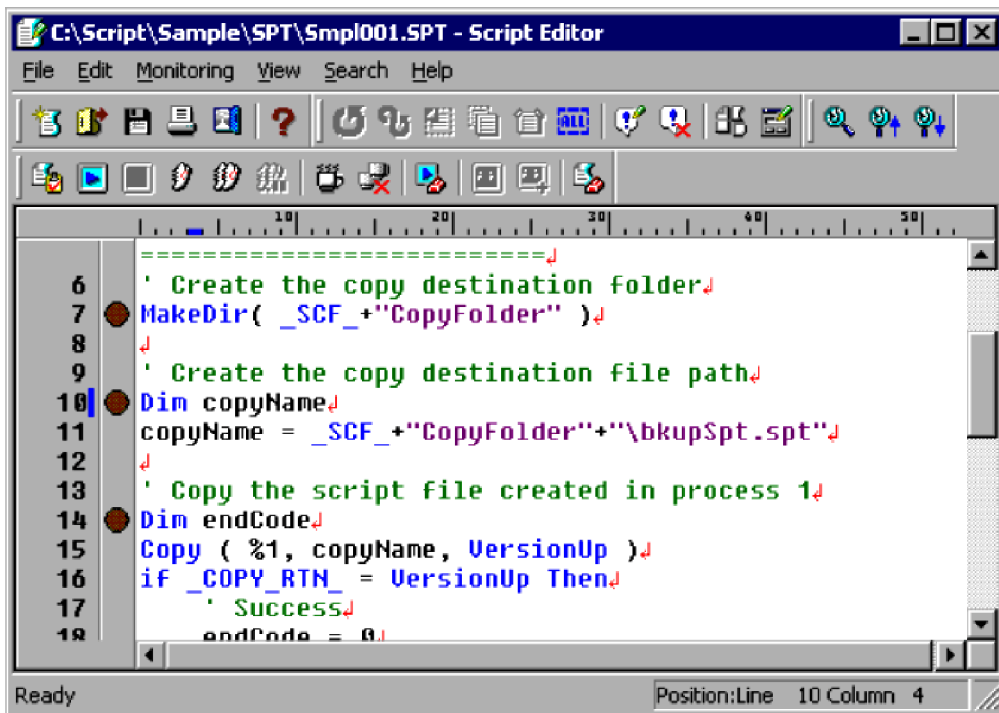
A breakpoint is set at the line where the cursor is located. The breakpoint is indicated by a circle symbol (●) to the left of the line.



- Canceling a breakpoint

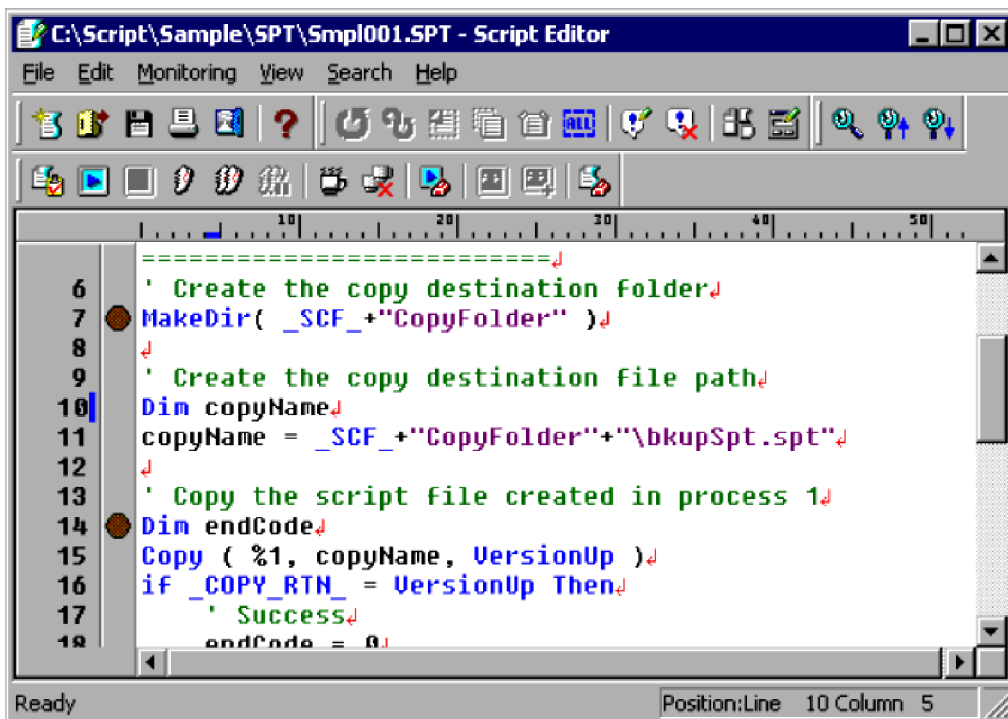
To cancel a breakpoint:

1. Move the cursor to the line at which you want to remove the breakpoint.



2. Choose **Monitoring, Cancel Breakpoint**.

The breakpoint is canceled at the line where the cursor is located.



- Canceling all breakpoints

To cancel all breakpoints:

1. Choose **Monitoring, Cancel All Breakpoints**.

```

7 MakeDir( _SCF_+"CopyFolder" )
8
9 ' Create the copy destination file path
10 Dim copyName
11 ● copyName = _SCF_+"CopyFolder"+"\bkupSpt.spt"
12
13 ' Copy the script file created in process 1
14 ● Dim endCode
15 Copy ( %1, copyName, VersionUp )
16 if _COPY_RTN_ = VersionUp Then
17     ' Success
18     endCode = 0
19 elseif _COPY_RTN_ = Skip Then
20     ' Skipped because the date is the same

```



```

7 MakeDir( _SCF_+"CopyFolder" )
8
9 ' Create the copy destination file path
10 Dim copyName
11 copyName = _SCF_+"CopyFolder"+"\bkupSpt.spt"
12
13 ' Copy the script file created in process 1
14 Dim endCode
15 Copy ( %1, copyName, VersionUp )
16 if _COPY_RTN_ = VersionUp Then
17     ' Success
18     endCode = 0
19 elseif _COPY_RTN_ = Skip Then
20     ' Skipped because the date is the same

```

All breakpoints in the displayed script file are canceled.

Notes

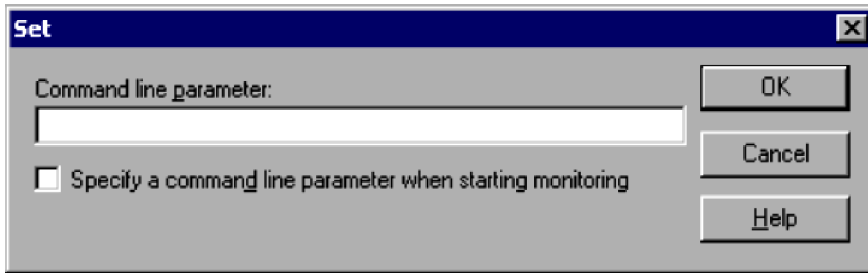
- In edit mode, you can set a breakpoint at any line. In monitoring mode, you can set breakpoints only for executable commands or statements in their entirety.
- If you set a breakpoint at a comment line or line space, Editor automatically searches downward for an appropriate position in the script and sets a breakpoint there. If no appropriate position can be found, no breakpoint is set. Instead, execution halts at the first line in the script file.

3.2.11 Setting the operating environment for monitoring mode

To set the operating environment for execution in monitoring mode:

1. Choose **Monitoring, Set**.

The Set dialog box appears.



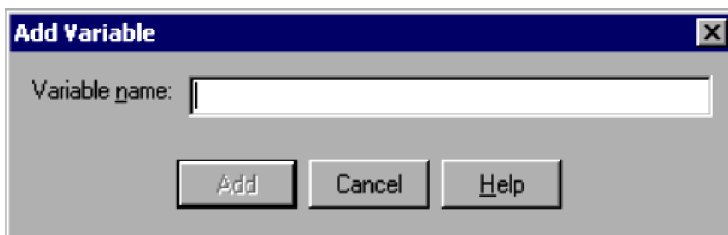
For details about using this dialog box, see [4.2.5 Set dialog box](#).

2. Enter the required information.
Set preferences for monitoring script execution.
3. Click the **OK** button.
The operating environment is set for monitoring mode and the dialog box closes.

3.2.12 Adding a variable to the Watch window

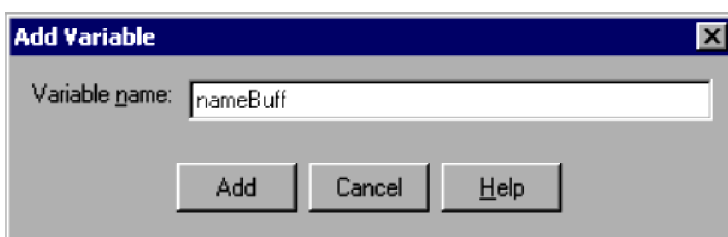
To add a specified variable to the Watch window:

1. In monitoring mode, choose **Monitoring, Add to Watch Window**.
The Add Variable dialog box appears.

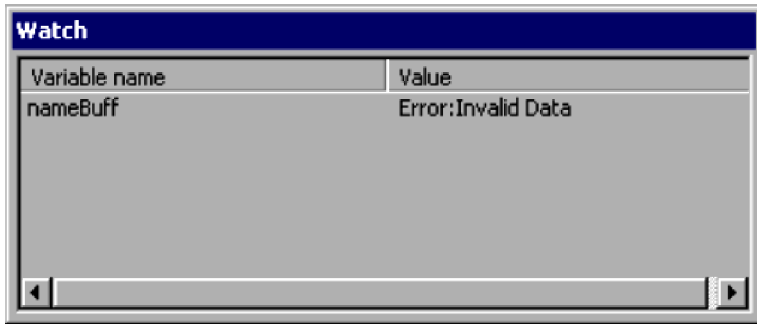


For details about using this dialog box, see [4.2.10 Add Variable dialog box](#).

2. Enter the name of the variable you want to add.
Up to 99 characters can be specified. You cannot enter variable names over 99 characters.



3. Click the **Add** button.
The variable name you entered is added to the Watch window.



Notes

- If you select a character string in the client area of the Script Editor window while in monitoring mode and then choose **Monitoring, Add to Watch Window**, the selected string appears as a variable name in the Watch window.
- Click the **Cancel** button to close the Add Variable dialog box without adding a variable.
- To delete an added variable from the Watch window, right-click on the variable name and choose **Delete Variable**.

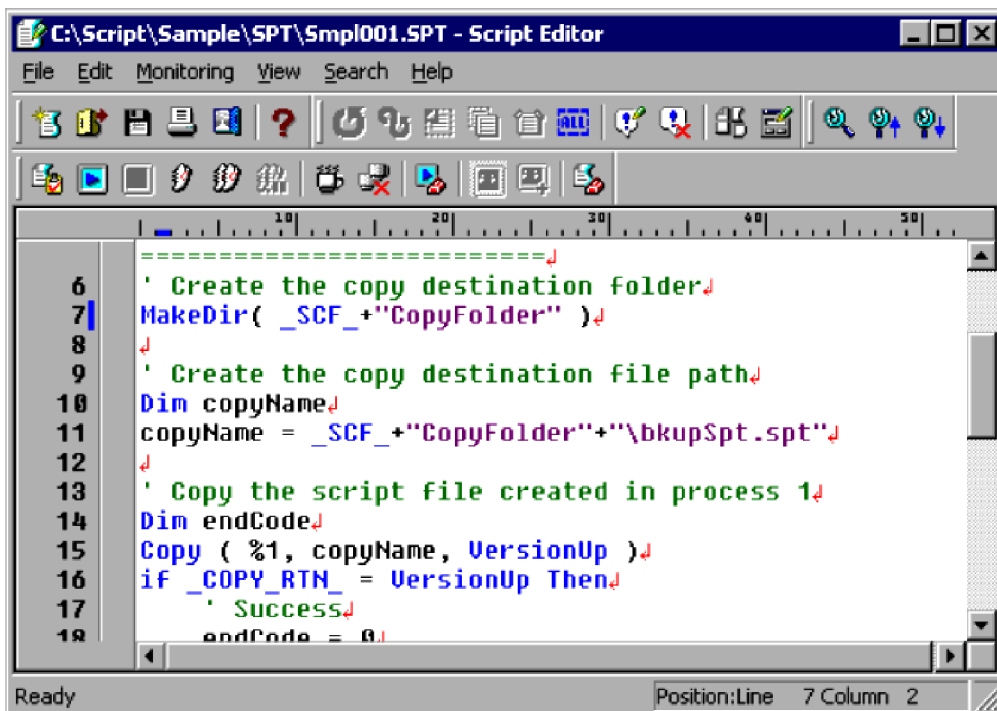
3.2.13 Setting the script execution environment

The following describes how to set the execution environment for a created script file. You can also do this from the Script Manager window.

To set the execution environment for multiple script files, use the Script Manager window. For details about how to use the Script Manager window, see [3.1.12 Setting the script execution environment \(all items\)](#).

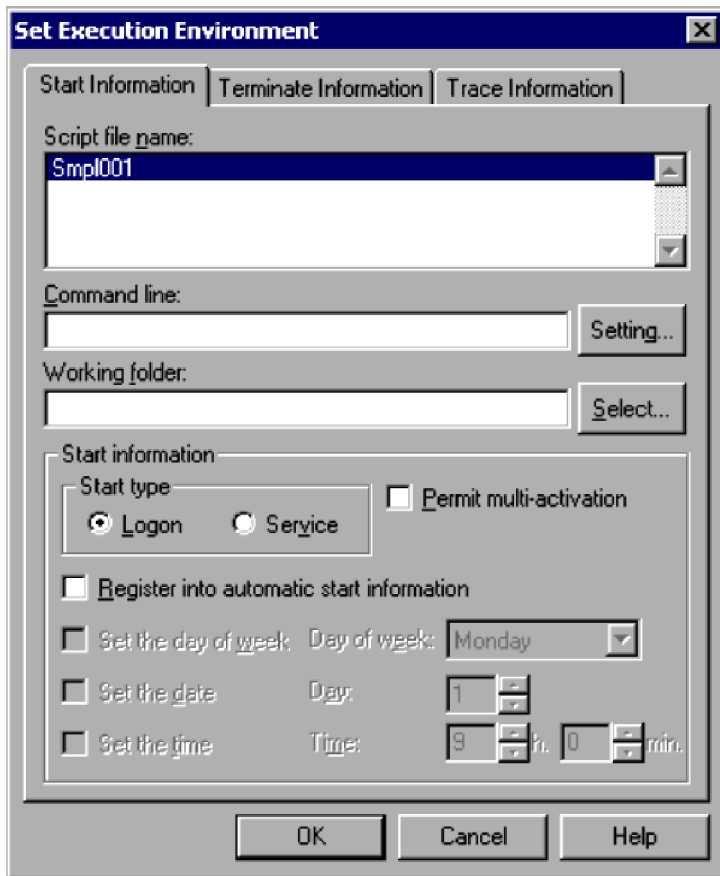
To set the script execution environment from Script Editor:

1. Display the script file for which you want to set the execution environment.



2. Choose **Monitoring, Set Execution Environment**.

The Set Execution Environment (Start Information) dialog box appears.



For details about using this dialog box, see [4.1.8 Set Execution Environment \(Start Information\) dialog box](#).

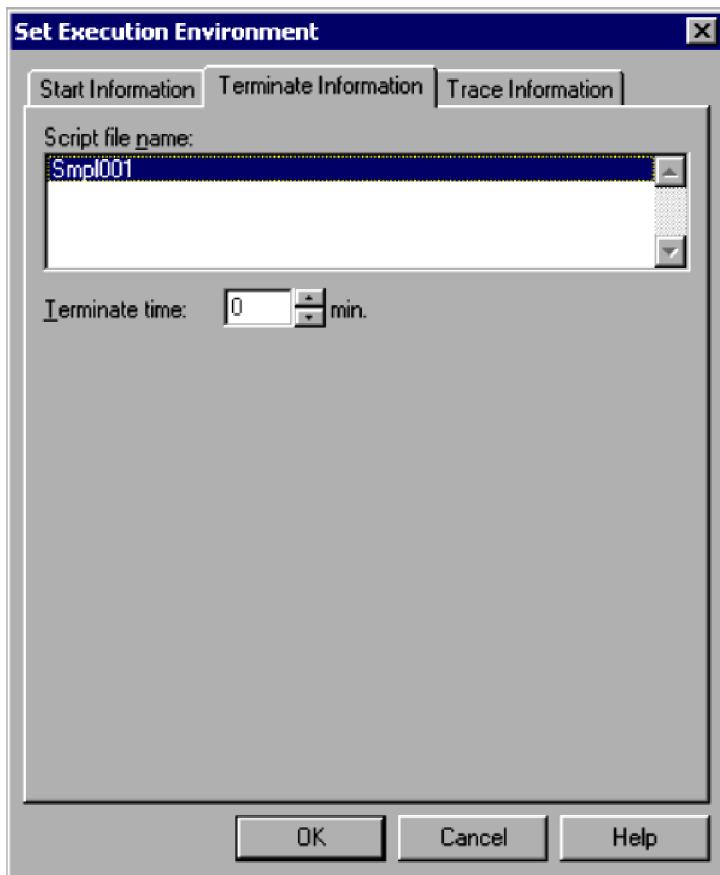
3. Set the required start information.

You must write a command line if a parameter is required for processing, or if you wish to set the output format of the analysis trace or execution trace.

For details on specifying command lines, see [6.2 Rules for writing command lines](#).

4. Click the **Terminate Information** tab.

The Set Execution Environment (Terminate Information) dialog box appears.



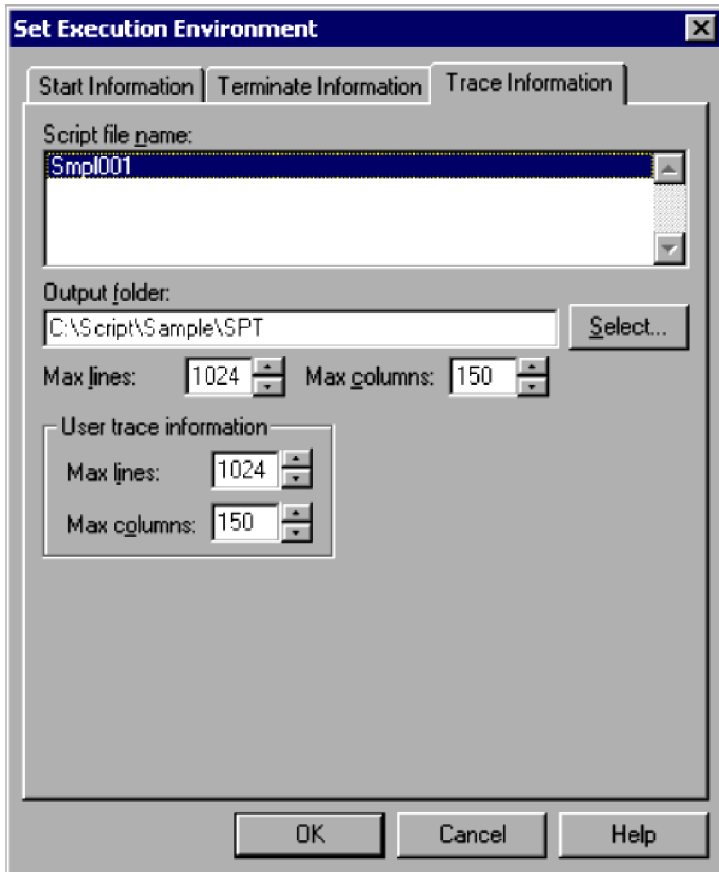
For details about using this dialog box, see [4.1.9 Set Execution Environment \(Terminate Information\) dialog box](#).

5. Set the termination time.

If you want to disable forced termination at a set time, make sure the time is set to zero.

6. Click the **Trace Information** tab.

The Set Execution Environment (Trace Information) dialog box appears.



For details about using this dialog box, see [4.1.10 Set Execution Environment \(Trace Information\) dialog box](#).

7. Set the required trace information.

You do not need to set any information in this dialog box unless you particularly want to set the number of output lines in the trace file or other such information.

8. Click the **OK** button.

The execution environment is set and the dialog box closes.

Note

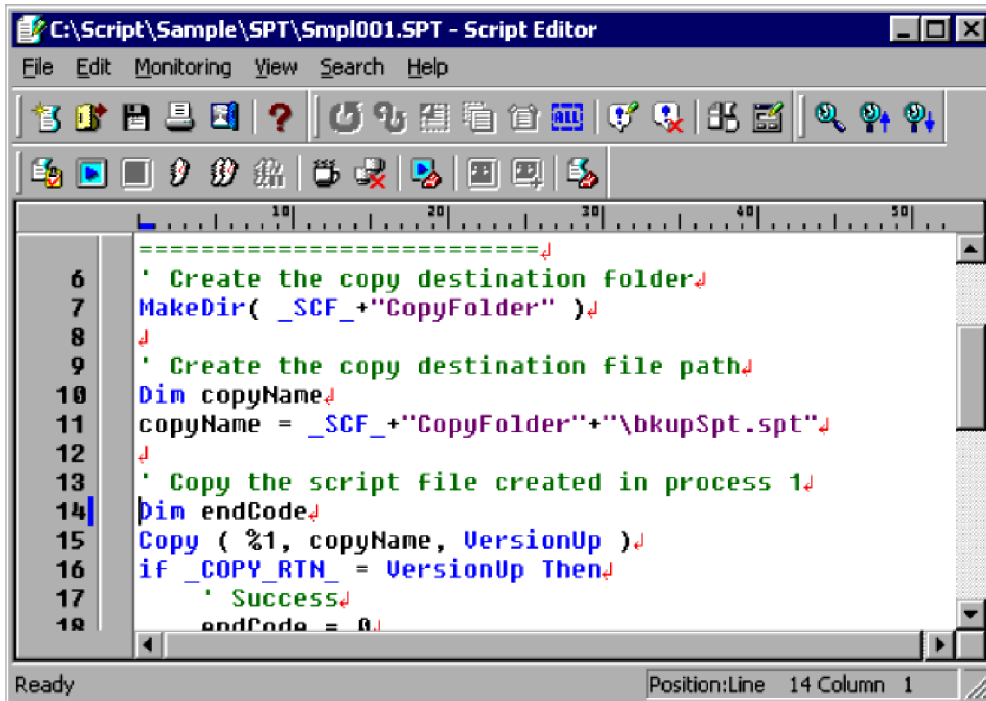
If you attempt to set up a script execution environment while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

3.2.14 Finding text in a file

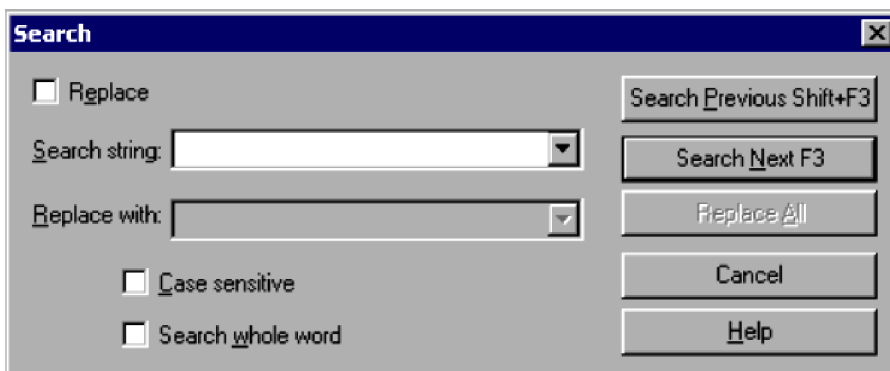
To search for text in a script file:

1. Display the script file that you want to search.



2. Choose **Search, Find**.

The Search dialog box appears.

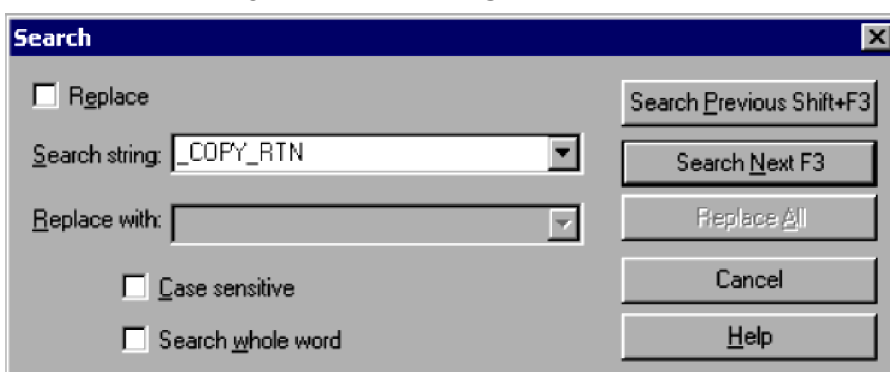


For details about using this dialog box, see [4.2.6 Search dialog box](#).

3. Make sure the **Replace** check box is not selected.

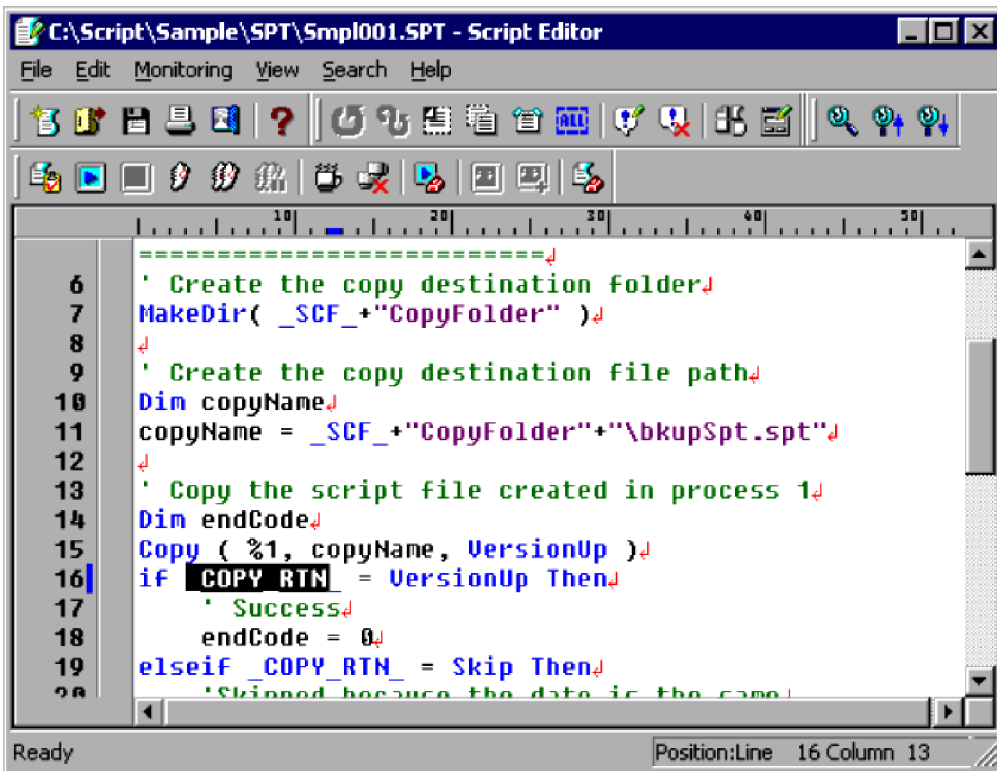
Clear the **Replace** check box if selected.

4. Enter the search string in the **Search string** box. Select the **Case sensitive** and **Search whole word** boxes as required.



5. Click the **Search Previous** or **Search Next** button.

JP1/Script looks for the search string. A beep sounds if there is no such text in the file.



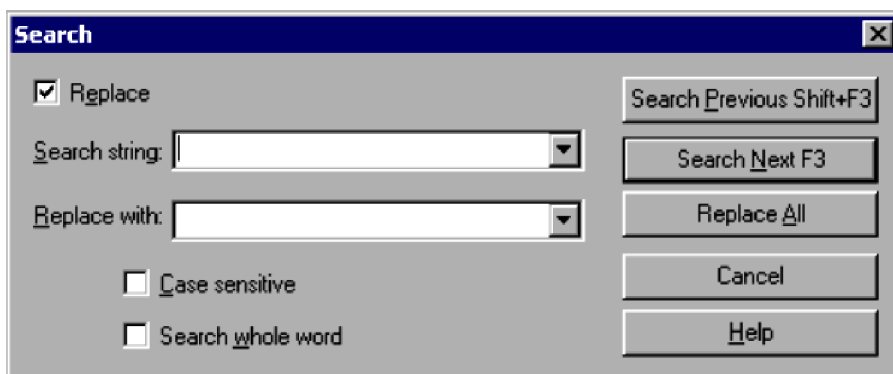
6. Click **Cancel** to finish searching.
The Search dialog box closes.

3.2.15 Replacing text in a file

To replace text in a script file:

1. Display the script file in which you want to replace text.
2. Choose **Search, Replace**.

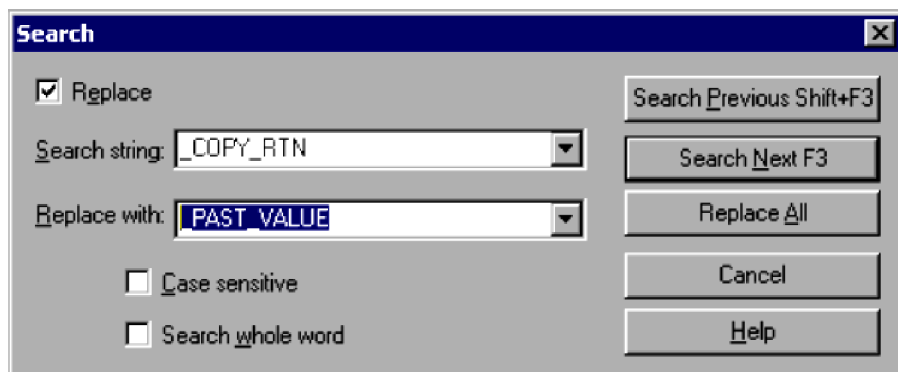
The Search dialog box appears.



For details about using this dialog box, see [4.2.6 Search dialog box](#).

3. Make sure the **Replace** check box is selected.
Click the **Replace** check box if not already selected.

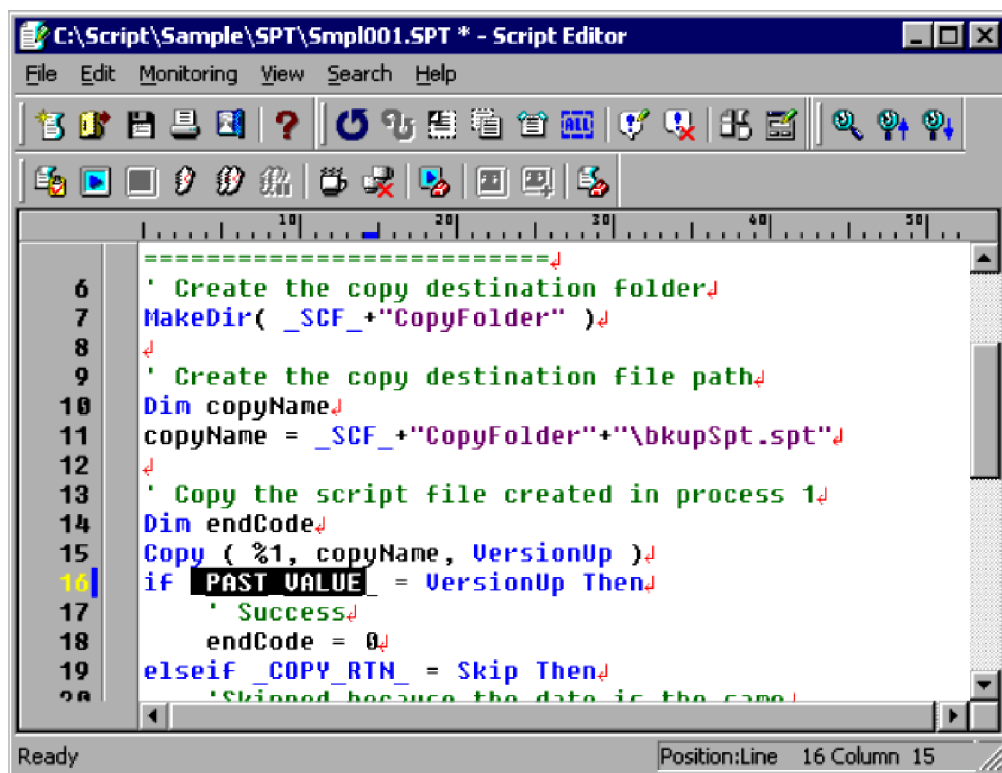
4. In the **Search string** box, enter the text to be replaced. Select the **Case sensitive** and **Search whole word** boxes as required.



5. In the **Replace with** box, enter the replacement text.

6. Click the **Search Previous** or **Search Next** button.

JP1/Script starts replacing text as specified. A beep sounds if there is no such text to be replaced in the file.



7. Click **Cancel** to finish replacing text.

The Search dialog box closes.

3.3 Easy Input operations

JP1/Script simplifies the task of entering commands and statements. Using the Easy Input program, you can enter commands easily and accurately without needing any command-specific knowledge. The commands and statements you enter are pasted to the clipboard.

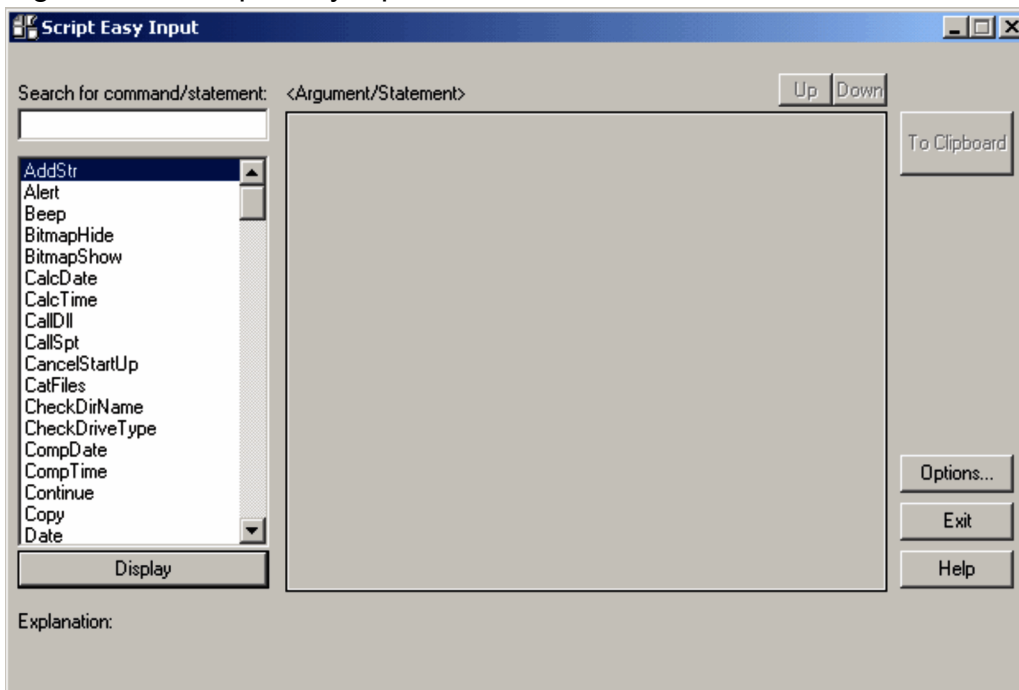
3.3.1 Starting Easy Input

Use one of the following methods to display the Script Easy Input window:

- Double-click the Easy Input icon.
- In the Script Editor window, choose **Edit, Easy Input**.
- In the Manager window, choose **Tools, Start Easy Input**.

Figure 3-3 shows the Script Easy Input window.

Figure 3–3: Script Easy Input window



(1) Processing target version

Shows the version of the specified Script Engine. The **Search for command/statement** list box contains only the commands and statements for that version.

(2) Search for command/statement

Lists the commands and statements for the specified version of Script Engine. You can locate a command or statement in this list by entering all or part of its name.

(3) Explanation

Gives a brief explanation of a command or statement selected in the list box.

(4) Target OS

Shows the operating system that supports the selected command or statement.

(5) Argument/Statement

Shows the arguments that can be entered in a selected command. When a statement is selected in the list box, this area displays the statement syntax.

The Script Easy Input window has the following buttons:

Display

Shows the arguments of a command, or syntax of a statement, selected in the list box. If the selected command has no arguments, a message to that effect is displayed.

To Editor or To Clipboard

The button reads **To Editor** when you access the Easy Input facility from Editor. The button reads **To Clipboard** when you access the Easy Input facility from the Manager or from the Windows **Start** menu.

Choose this button to paste the displayed command arguments or statement syntax to Editor or the clipboard.

Options

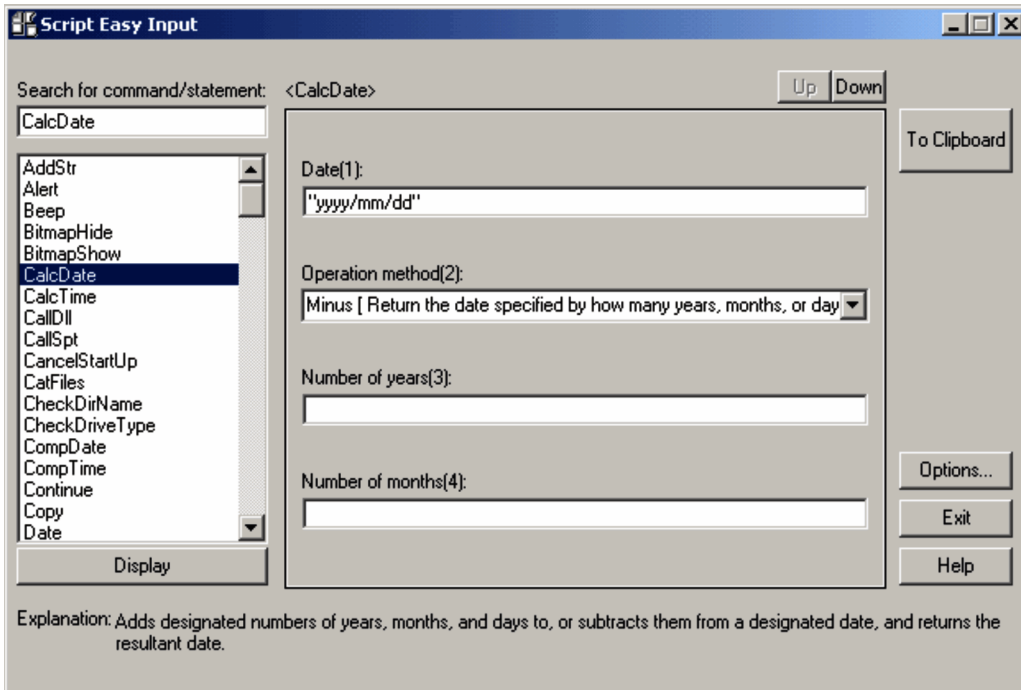
Displays the Easy Input Options (General) dialog box. For details about using this dialog box, see [3.3.3 Specifying the JP1/Script version for script creation](#).

Exit

Closes the Script Easy Input window.

3.3.2 Entering commands and statements

To write a command or statement when creating a script, use the Script Easy Input window. The following example is based on the `CalcDate` command.



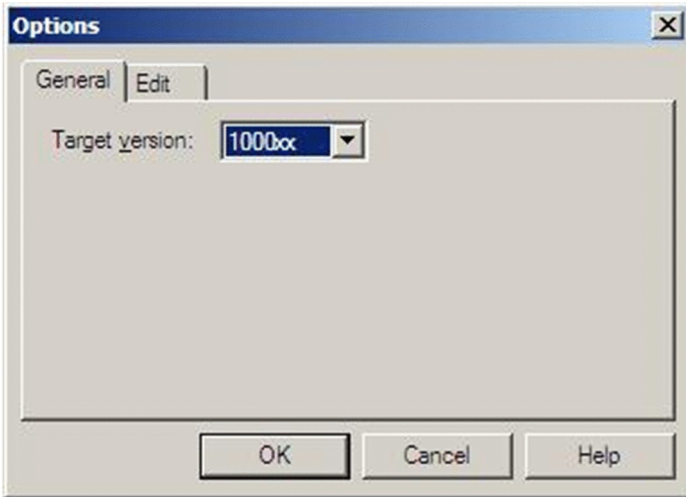
(1) Operations

- **Display** button
Displays the arguments that need to be set in the selected command, or displays the statement syntax if you selected a statement, in the **Argument/Statement** area. If you selected a command without any arguments, a message to that effect appears in this area.
- Enter command arguments if necessary.
- **To Clipboard** button
Pastes the contents set in the **Argument/Statement** area to the clipboard.
- **Exit** button
Closes the Script Easy Input window.

3.3.3 Specifying the JP1/Script version for script creation

In the Script Easy Input window, click the **Options** button to display the Options dialog box.

This dialog box has two pages: **General** and **Edit**. Click the **General** tab to display the Options (General) dialog box.



(1) Items

Target version

Set the JP1/Script version for script file creation. The default is the version of the installed JP1/Script.

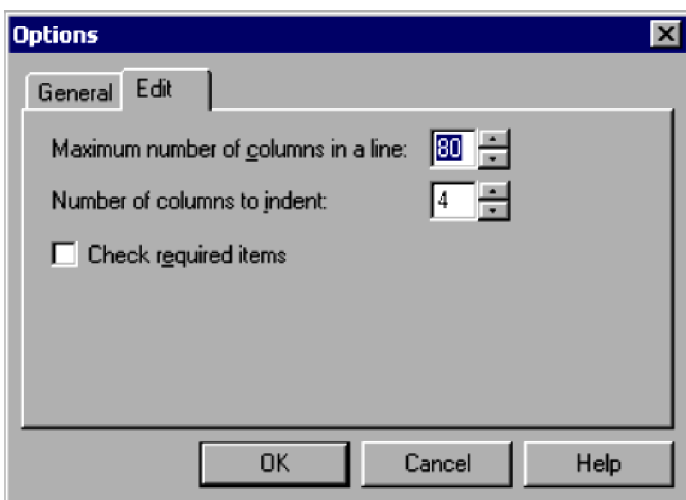
(2) Operations

- **OK** button
Applies the setting and closes the Options dialog box.
- **Cancel** button
Closes the dialog box without changing the original setting.

3.3.4 Setting the line length and indent size

In the Script Easy Input window, click the **Options** button to display the Options dialog box.

This dialog box has two pages: **General** and **Edit**. Click the **Edit** tab to display the Options (Edit) dialog box.



(1) Items

Maximum number of columns in a line

Specify the maximum number of columns per line in the edited contents pasted to Script Editor or the clipboard.

Specify a number in the range from 64 to 999. 80 is set by default.

Number of columns to indent

Specify the indent size of the edited contents pasted to the clipboard.

Specify a number in the range from 0 to 16. 4 is set by default.

Check required items

Select this check box to check whether data is correctly entered in the mandatory arguments of the commands you enter.

(2) Operations

- **OK** button
Applies the setting and closes the Options dialog box.
- **Cancel** button
Closes the dialog box without changing the original settings.

3.4 Trace Viewer operations

Trace Viewer is used to display the execution status and trace information for scripts created in Manager. The execution status and trace information are output to a trace file.

Note

If you attempt to start Trace Viewer while the JP1/Script service is not running, a dialog box containing the following warning appears and Trace Viewer startup is canceled:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

3.4.1 Script Trace Viewer window and menus

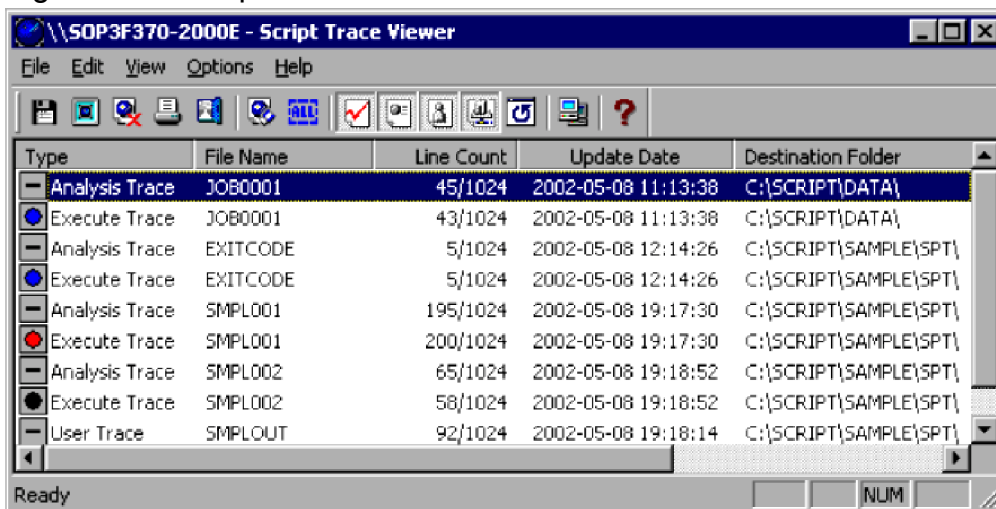
Trace Viewer displays the execution status and trace of a script created in Manager. The execution status and trace information are output to a trace file.

This section describes the layout and menus of the Script Trace Viewer window which appears when you start Trace Viewer. The window displayed when you open a trace file is also described.

(1) Script Trace Viewer window

Figure 3-4 shows the Script Trace Viewer window and the names of its components.

Figure 3–4: Script Trace Viewer window



(a) Toolbar

The toolbar contains buttons representing the most frequently used commands on the pull-down menus. By simply clicking a button, you can execute the corresponding command. You can hide the toolbar by toggling **Toolbar** in the **View** menu.

The following command buttons are displayed on the toolbar of the Script Trace Viewer window:

Save

Saves a trace file under a new name.

View

Displays the contents of a trace file.

Delete

Deletes a trace file.

Print

Prints a trace file.

Exit

Quits Trace Viewer.

Clear

Clears a trace file (to zero file size).

Select All

Selects all the items.

Analysis Trace

Shows or hides a list of analysis traces.

Execution Trace

Shows or hides a list of execution traces.

User Trace

Shows or hides a list of user traces.

Server Trace

Shows or hides a list of server traces.

Refresh

Updates the contents of a displayed trace file.

Select Computer

Specifies the computer on which the trace files to be displayed reside.

Help

Displays the JP1/Script online help.


(b) Status bar








The status bar displays messages about the processing being carried out by Trace Viewer and status messages at completion of processing.

(c) Client area

The client area displays trace file information (file type, file name, file size, date modified, and folder in which the file is located). In the leftmost field, icons show the execution status of each trace file. Their meaning differs depending on whether the particular script can be activated concurrently from multiple processes. Tables 3-11 and 3-12 show the meanings of the icons in each case.







Table 3–11: Script Trace Viewer icons when multi-activation is permitted

Icon#	Execution status	Meaning
 (white & lime)	Executing	One or more scripts is currently executing.

Icon#	Execution status	Meaning
 (white & blue)	Terminated normally (partially completed)	One or more scripts has terminated normally.
 (white & red)	System error (partially completed)	A command execution error occurred in one or more scripts. Script execution has been canceled.
 (white & yellow)	User error (partially completed)	An <code>Alert</code> command or <code>ExitWindows</code> command was executed in one or more scripts, or one or more scripts was forcibly terminated. (A system error may have occurred after <code>Alert</code> command execution.)
 (aqua & blue)	Terminated normally (all completed)	All scripts have terminated normally.
 (aqua & black)	Unexecuted (all completed)	None of the scripts has executed. This status is displayed when a command could not be executed because an error was found at syntax analysis, for example.
 (aqua & red)	System error (all completed)	All scripts have terminated, but a command execution error occurred in one or more scripts and script execution was canceled.
 (aqua & yellow)	User error (all completed)	All scripts have terminated, but an <code>Alert</code> command or <code>ExitWindows</code> command was executed in one or more scripts, or one or more scripts was forcibly terminated. (A system error may have occurred after <code>Alert</code> command execution.)

#: The colors in parentheses in the Icon field are the colors of the circles in the icon.

Table 3–12: Script Trace Viewer icons when multi-activation is prohibited

Icon#	Execution status	Meaning
	Not applicable	Execution status is not managed for this type of file (analysis trace or user trace).
 (black)	Unexecuted	The script has not executed.
 (lime)	Executing	The script is executing.
 (blue)	Terminated normally	The script has terminated normally.
 (red)	Terminated abnormally (system error)	A command execution error occurred and script execution has been canceled.
 (yellow)	Terminated abnormally (user error)	An <code>Alert</code> command or <code>ExitWindows</code> command was executed or the script was forcibly terminated. (A system error may have occurred after <code>Alert</code> command execution.)

#: The color in parentheses in the Icon field is the color of the circle in the icon.

(2) Menus in the Script Trace Viewer window

The menus in the Script Trace Viewer window are described next. Table 3-13 lists the Trace Viewer commands (functions) provided in the menus.

Table 3–13: Trace Viewer menus and commands

Menu	Command (function)	Description
File	Save As	Saves a selected trace file under a new name.
	View Trace File	Displays the contents of a selected trace file.
	Delete Trace File	Deletes a selected trace file.
	Print	Prints a selected trace file.
	Exit	Quits Trace Viewer.
Edit	Clear Trace File	Clears a trace file (to zero file size).
	Select All	Selects all items.
View	Toolbar	Shows or hides the toolbar.
	Status Bar	Shows or hides the status bar.
	Minimize When Inactive	Minimizes the Trace Viewer window to an icon when the window is not in use.
	Analysis Trace	Shows or hides a list of analysis traces.
	Execution Trace	Shows or hides a list of execution traces.
	User Trace	Shows or hides a list of user traces.
	Server Trace	Shows or hides a list of server traces.
	Refresh	Updates the information in a trace file.
Options	Select Computer	Specifies the computer on which the trace files to be displayed reside.
Help	Contents	Displays the contents of the JP1/Script online help.
	Search by Keyword	Lists the keywords of the JP1/Script online help.
	About JP1/Script	Displays version information for JP1/Script.

3.4.2 Mouse and key operations in the Script Trace Viewer window

The mouse and key operations you can perform in the Script Trace Viewer window are described below.

(1) Mouse operations

Table 3-14 lists mouse operations in the client area of the Script Trace Viewer window.

Table 3–14: Mouse operations in the Script Trace Viewer window

Operation	Processing
Click	Clears the previous selection and makes a new selection.
Double-click	Displays the Trace Files Display window.

(2) Key operations

Table 3-15 lists key operations in the client area of the Script Trace Viewer window.

Table 3–15: Key operations in the Script Trace Viewer window

Operation	Processing
Ctrl + A	Selects all trace files.
Ctrl + P	Prints a trace file.
Del	Deletes a trace file.
Enter	Displays the Trace Files Display window.

3.4.3 Starting and stopping Trace Viewer

(1) Starting Trace Viewer

The trace produced when you execute a script file is output to a trace file. To check the trace status, start Trace Viewer.

There are two ways of starting Trace Viewer.

(a) Starting from the Windows Start menu

To start Trace Viewer:

1. From the Windows **Start** menu, choose **Programs, JP1/Script**, then **Trace Viewer**.
The Script Trace Viewer window appears, displaying a list of trace files.

Note

If you attempt to start Trace Viewer while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(b) Starting from the Script Manager window

To start Trace Viewer:

1. In the Script Manager window, choose **Tools**, and then **Start Trace Viewer**.
You can start Trace Viewer.

Note

If you attempt to start Trace Viewer while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(2) Stopping Trace Viewer

Choose **File, Exit**. Trace Viewer stops and the Script Trace Viewer window closes.

3.4.4 Operations in the Script Trace Viewer window

You can perform the following operations in the Script Trace Viewer window:

- Show or hide trace files
- Display trace file contents
- Save a trace file under a new name
- Print a trace file
- Delete a trace file
- Clear a trace file
- Search for text in a trace file
- Show or hide the toolbar[#]
- Show or hide the status bar[#]
- Minimize the Trace Viewer window when not in use[#]
- Refresh the client area[#]
- Specify the computer on which the trace files to be displayed reside[#]
- Close the Trace Files Display window[#]
- Show or hide line numbers in a trace file[#]
- Arrange Trace Files Display windows vertically or horizontally[#]

#

These operations are not described. They are either standard Windows operations or simply involve choosing a command from a menu.

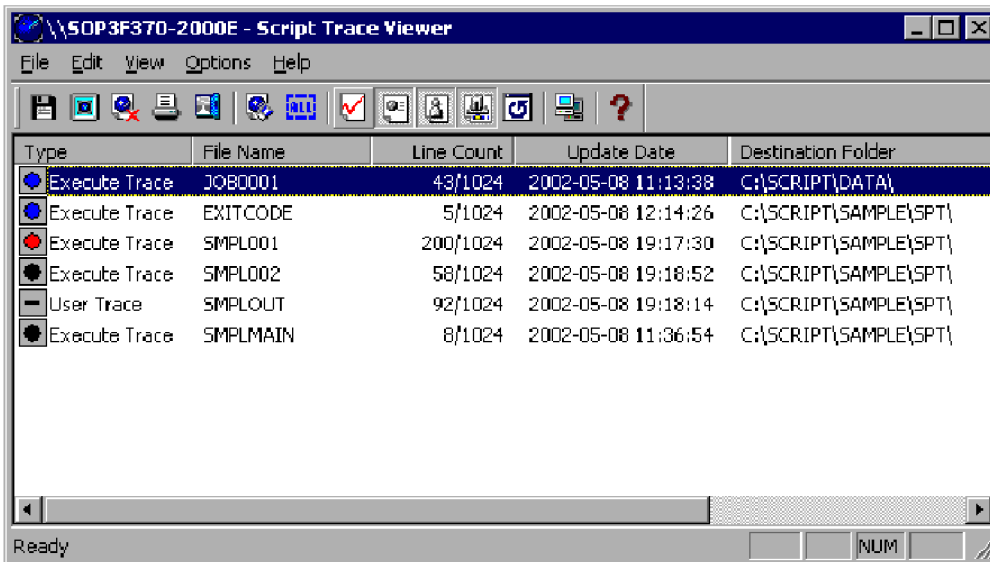
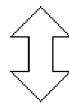
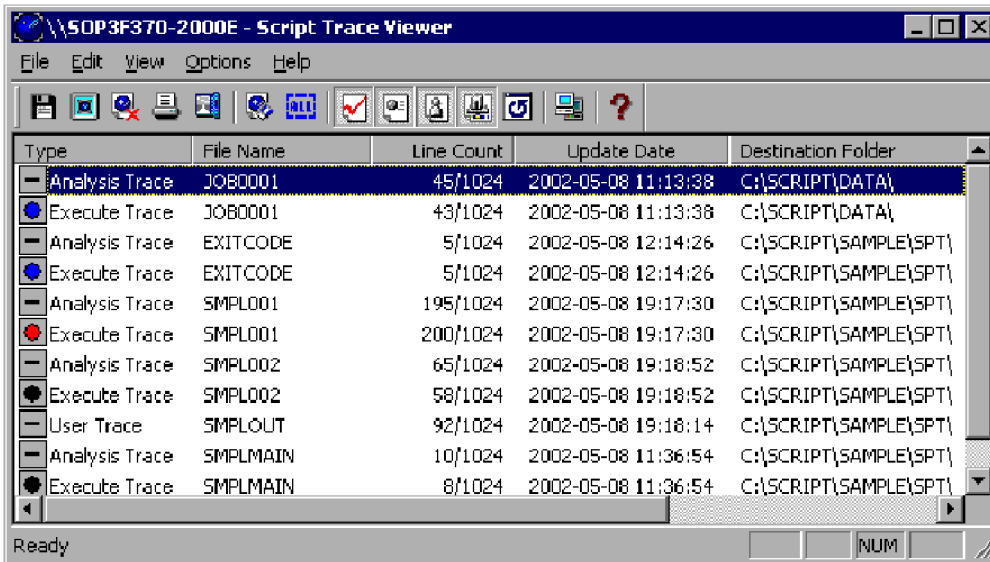
3.4.5 Showing or hiding trace files

You can toggle between showing or hiding the execution trace files, analysis trace files, user trace files, and server trace files listed in the Script Trace Viewer window. For example, when you are ready to execute a completed script file as a job, you can hide the analysis trace file since it is no longer needed.

Operation

1. To show or hide an execution trace, choose **View, Execution Trace**.
2. To show or hide an analysis trace, choose **View, Analysis Trace**.
3. To show or hide a user trace, choose **View, User Trace**.
4. To show or hide a server trace, choose **View, Server Trace**.

The following example illustrates toggling between showing and hiding an analysis trace.

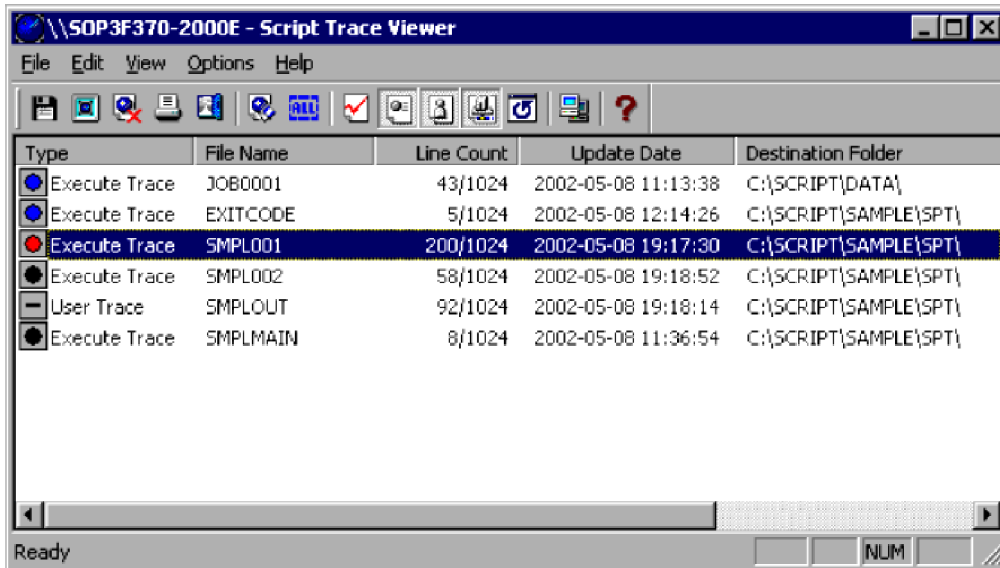


3.4.6 Displaying trace file contents

You can select a trace file in the Script Trace Viewer window and display its contents in a Trace Files Display window. You can view the contents of multiple trace files at the same time.

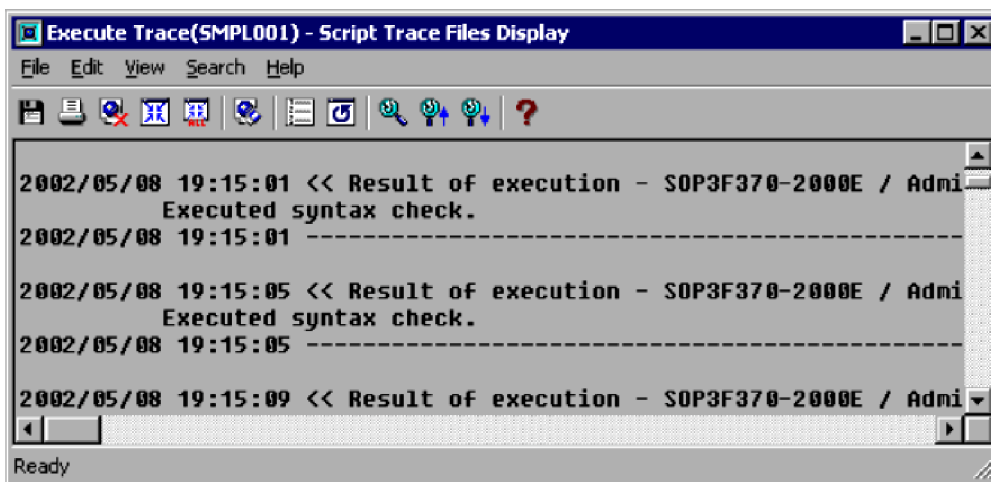
To view trace file contents:

1. In the Script Trace Viewer window, select one or more trace files.



2. Choose **File, View Trace File**.

The contents of the trace file appear in the Script Trace Files Display window.



3. In the Script Trace Files Display window, choose **File**, and then **Close View Window** or **Close All View Windows**.

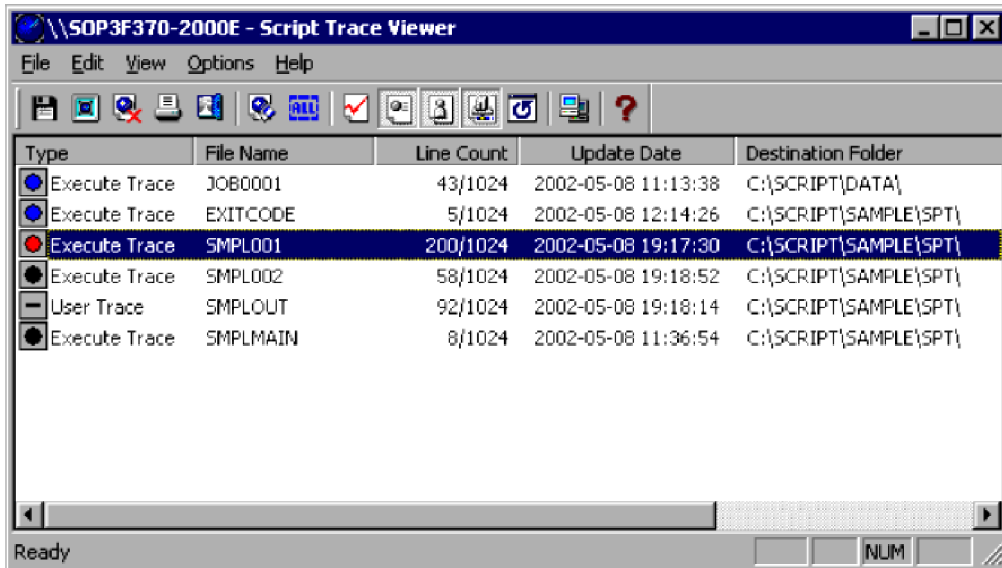
Note

If you have already chosen **View, Minimize When Inactive** in the Script Trace Viewer window, the Script Trace Viewer window appears as an icon when the Trace Files Display window opens at step 2.

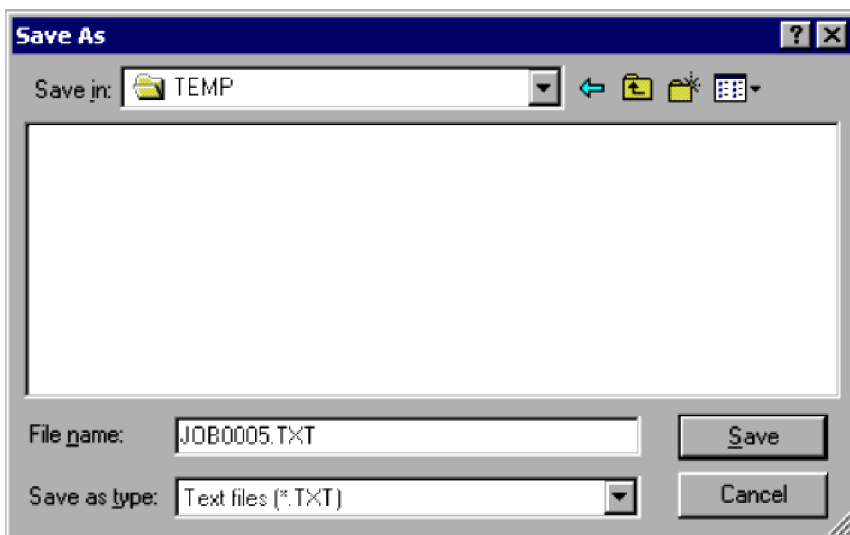
3.4.7 Saving a trace file under a new name

To save a selected trace file under a new name:

1. In the client area of the Script Trace Viewer window, select the trace file you want to save under a new name. Alternatively, make sure the trace file you want to save is currently displayed in a Trace Files Display window.



2. Choose **File, Save As**. In the displayed dialog box, enter the file name, then click **Save**.



The selected trace file is saved under the specified name.

Note

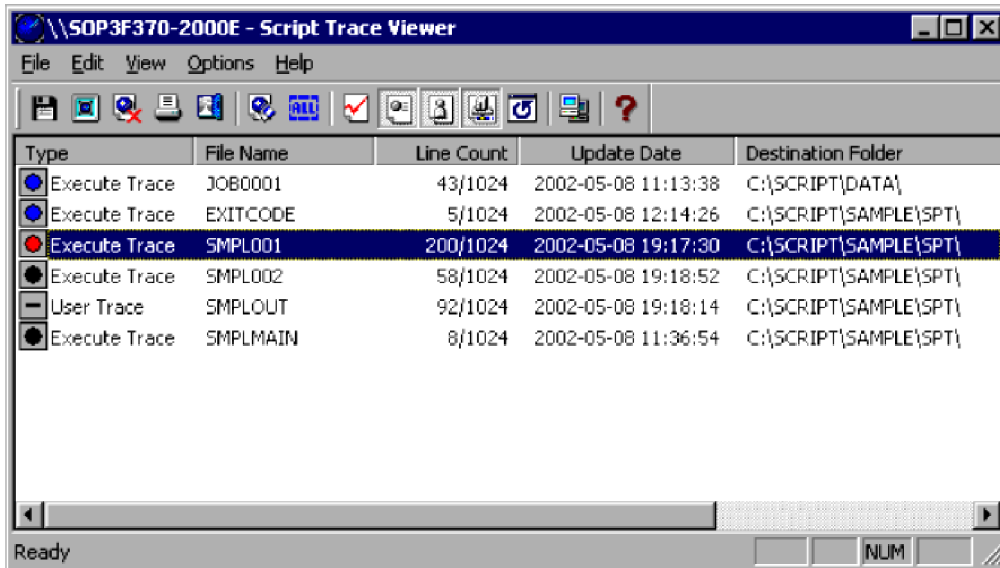
If you choose **Cancel** at step 2, in the Trace Viewer window, the Trace Viewer window appears as an icon when the Trace Files Display window opens at step 2.

3.4.8 Deleting a trace file

You can delete a trace file selected in the Script Trace Viewer window or displayed in a Trace Files Display window.

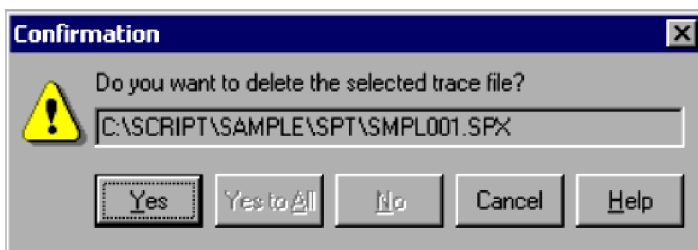
To delete a trace file:

1. In the client area of the Script Trace Viewer window, select one or more trace files that you want to delete. Alternatively, make sure the trace file(s) you want to delete are currently displayed in Trace Files Display windows.



2. Choose **File, Delete Trace File**.

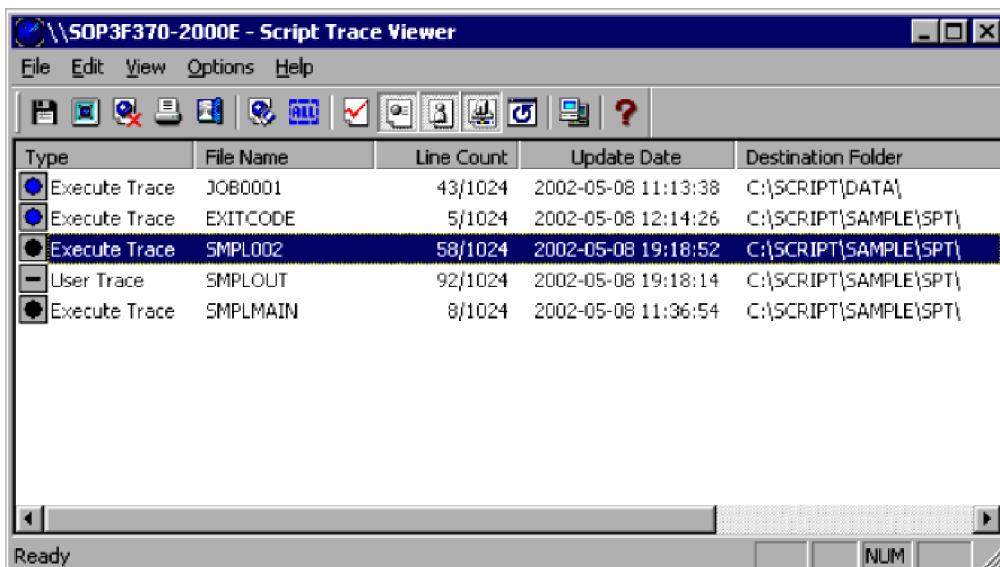
The Confirmation dialog box appears.



For details about using this dialog box, see [4.3.1 Confirmation \(delete trace file\) dialog box](#).

3. Click the **Yes** button.

One trace file is deleted. If you selected more than one trace file, you are then asked to confirm deletion of the next file.



This process is repeated for all the trace files you selected. When all the files have been deleted, the dialog box closes.

Notes

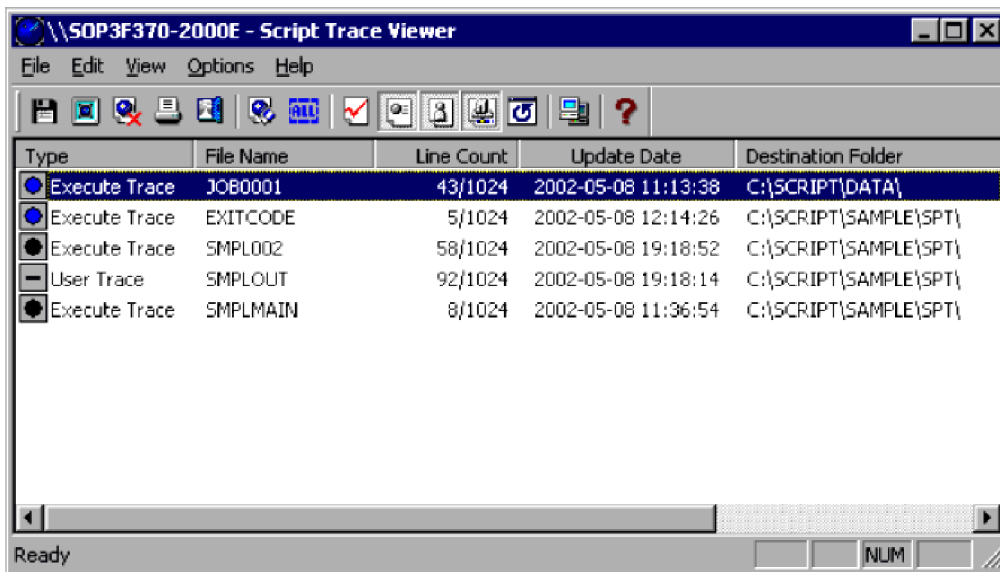
- If you click **Yes to All** in step 3, all the selected files are deleted.
- If you click **Cancel** in step 3, file deletion is canceled and the dialog box closes.

3.4.9 Clearing a trace file

You can clear a trace file selected in the Script Trace Viewer window or displayed in a Trace Files Display window. This operation reduces the file to zero size.

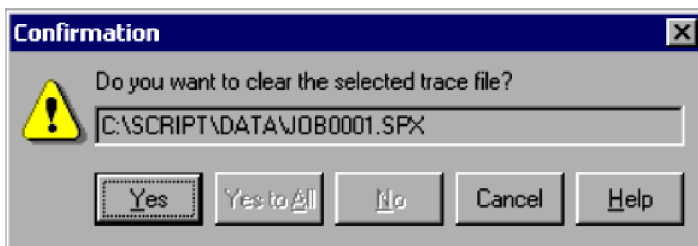
To clear a trace file:

1. In the client area of the Script Trace Viewer window, select one or more trace files that you want to clear. Alternatively, make sure the trace file(s) you want to clear are currently displayed in Trace Files Display windows.



2. Choose **Edit, Clear Trace File**.

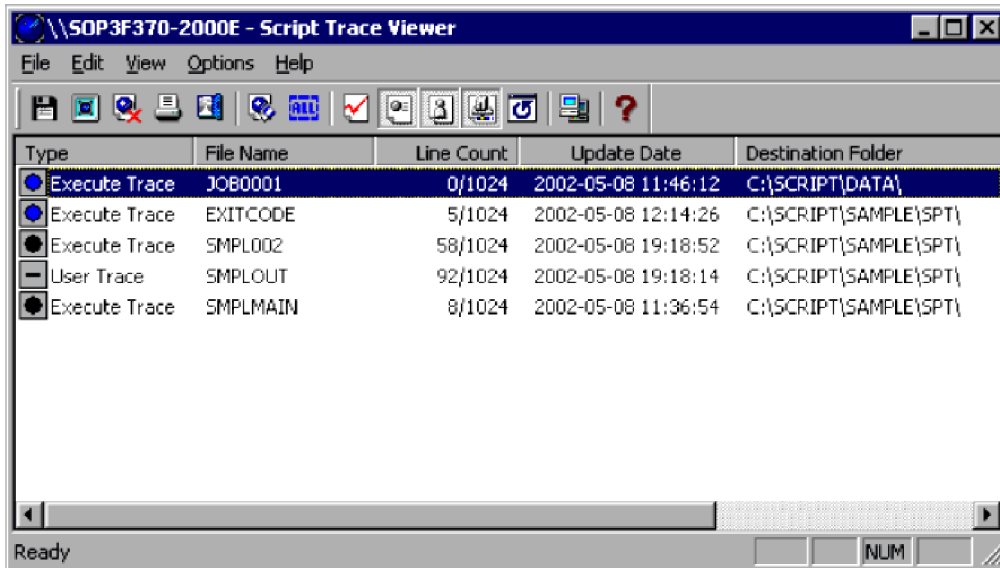
The Confirmation dialog box appears.



For details about using this dialog box, see [4.3.2 Confirmation \(clear trace file\) dialog box](#).

3. Click the **Yes** button.

One trace file is cleared.



If you selected more than one trace file, you are asked to confirm that you want to clear the next file.

This process is repeated for all the trace files you selected. When all the files have been cleared, the dialog box closes.

Notes

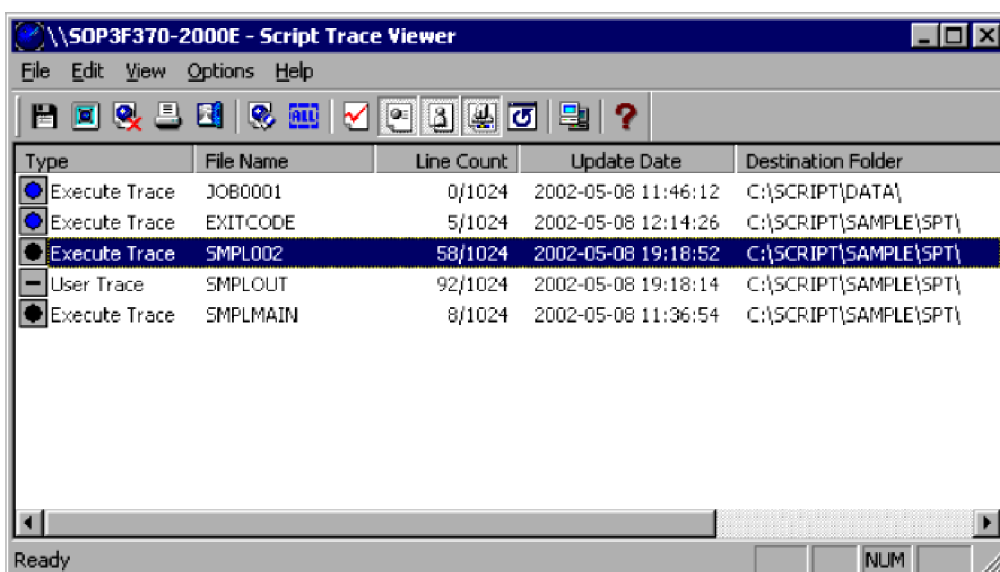
- If you click **Yes to All** in step 3, all the selected files are cleared.
- If you click **Cancel** in step 3, the operation is canceled and the dialog box closes.

3.4.10 Printing a trace file

You can print the contents of a trace file selected in the Script Trace Viewer window or displayed in a Trace Files Display window.

To print a trace file:

1. In the client area of the Script Trace Viewer window, select the trace file that you want to print. Alternatively, make sure the trace file you want to print is currently displayed in a Trace Files Display window.



2. Choose **File, Print**. Then enter the required information in the displayed Print dialog box.

3. Click the **OK** button.

The trace file is printed.

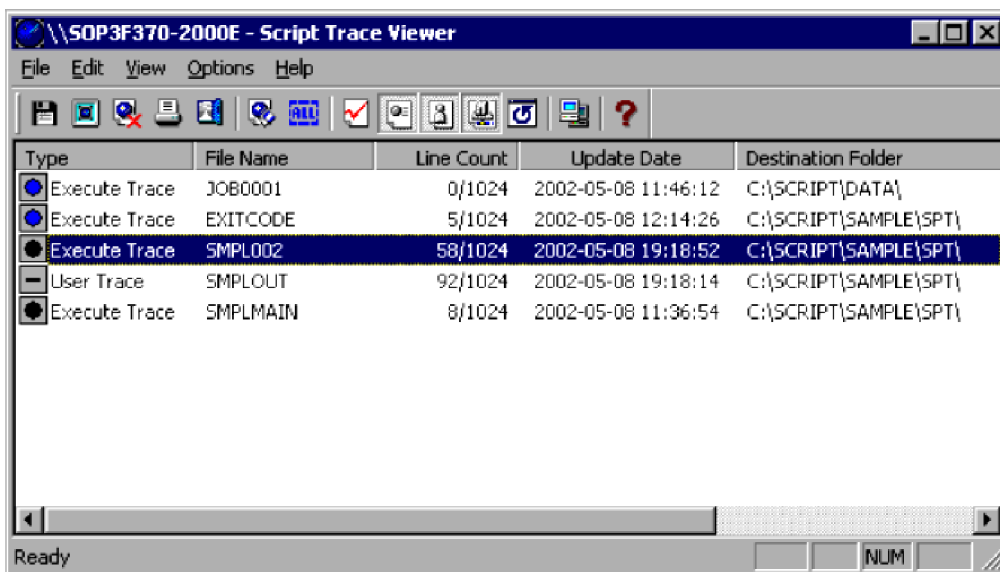
Notes

- If you click **Cancel** at step 3, printing is canceled and the dialog box closes.
- Tabs are assumed to be four spaces.
- If the file contents do not fit within the specified paper size, lines are wrapped to subsequent pages.

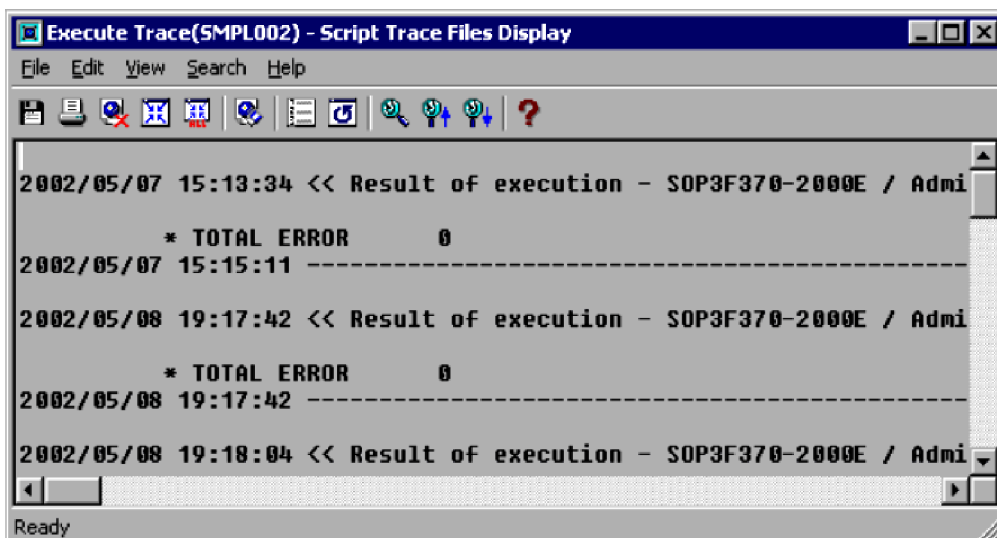
3.4.11 Finding text in a trace file

To search for a specified string in the active trace information:

1. In the client area of the Script Trace Viewer window, select the trace file that you want to search.



2. Choose **File, View Trace File** to display the trace file contents.



3. Choose **Search, Find**.

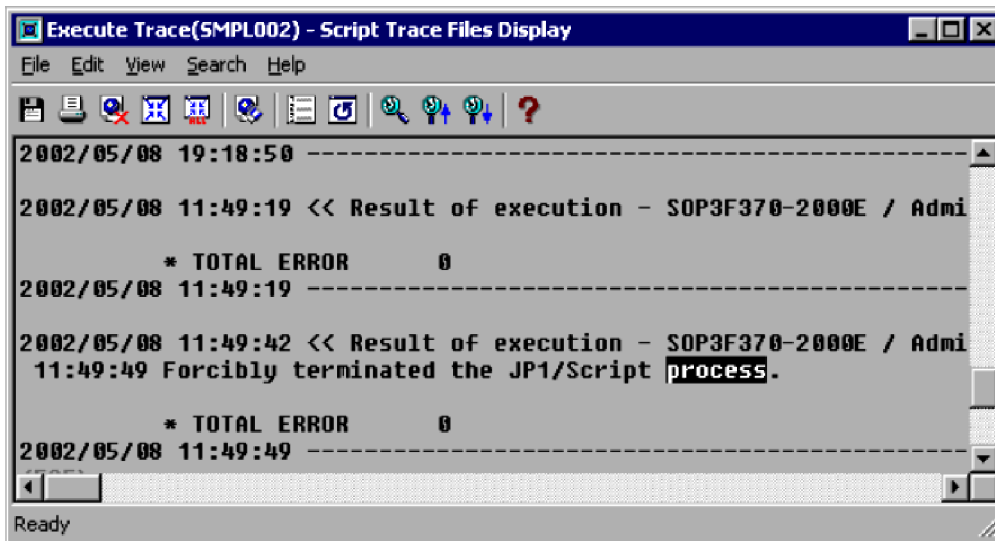
The Search dialog box appears.



For details about using this dialog box, see [4.3.4 Search dialog box](#).

4. Enter the search string in the Search box, then click the **Search Up** or **Search Down** button.

The search is performed.



Notes

- If you click **Cancel** at step 3, the dialog box closes and the search is not performed.
- A beep sounds if you attempt to execute a search without specifying a search string.

3.5 Trace Files Display operations

3.5.1 Trace Files Display window and menus

The Trace Files Display window displays the execution status and trace of an executed script.

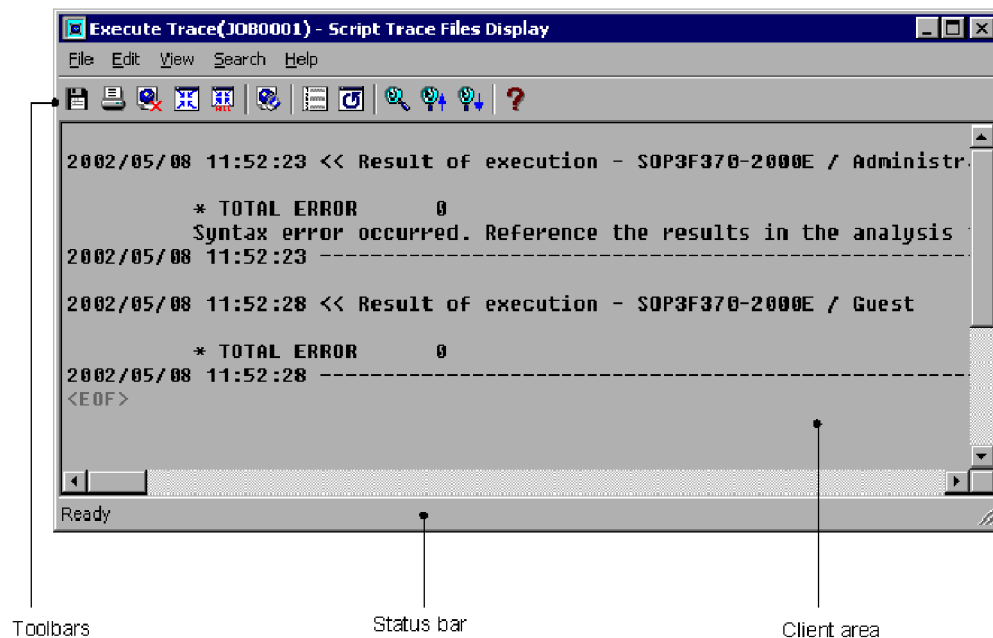
This section describes the layout and menus of the Trace Files Display window which appears when you open a trace file.

For information about the output formats of Trace Viewer files, see *A. Output Formats of Script Trace Files*.

(1) Script Trace Files Display window

Double-clicking a trace file displayed in the Script Trace Viewer window displays the Script Trace Files Display window. Figure 3-2 shows the Script Trace Files Display window and the names of its components.

Figure 3–5: Script Trace Files Display window



(a) Toolbar

The toolbar contains buttons representing the most frequently used commands on the pull-down menus. By simply clicking a button, you can execute the corresponding command. You can hide the toolbar by toggling **Toolbar** in the **View** menu.

The following command buttons are displayed on the toolbar of the Script Trace Files Display window:

Save

Saves a trace file under a new name.

Print

Prints a trace file.

Delete

Deletes a trace file.

Close

Closes the Trace Files Display window.

Close All

Closes all Script Trace Files Display windows that are open.

Clear

Clears a trace file (to zero file size).

Line Number

Displays or hides line numbers.

Refresh

Updates the contents of a trace file.

Find

Sets search criteria.

Search Up

Searches from the current position to the beginning of the file.

Search Down

Searches from the current position to the end of the file.

Help

Displays the JP1/Script online help.

(b) Status bar

The status bar displays messages about the processing being carried out by Trace Viewer and status messages at completion of processing.

(c) Client area

The client area displays the contents of a trace file.

A scroll bar appears if the contents do not fit within the visible client area.

You can change the font and font size for the characters that are displayed in this client area.

Registry key

HKEY_CURRENT_USER\SOFTWARE\HITACHI\JP1/Script\SPTTM\TraceWindow

Value name 1

Font

Value datatype 1

REG_SZ

Value 1

Font

When the setting takes effect

The setting takes effect the next time the Script Trace Files Display window opens.

Value name 2

FontSize

Value datatype 2

REG_DWORD

Value 2

FontSize

When the setting takes effect

The setting takes effect the next time the Script Trace Files Display window opens.

(2) Menus in the Script Trace Files Display window

This subsection describes the functions of the Script Trace Files Display window by menu. Table 3-16 lists and describes the menus provided in the Script Trace Files Display window.

Table 3–16: Menus in the Script Trace Files Display window

Menu	Command (function)	Description
File	Save As	Saves a selected trace file under a new name.
	Delete Trace File	Deletes a selected trace file.
	Print	Prints a selected trace file.
	Close View Window	Closes the active Trace Files Display window.
	Close All View Windows	Closes all the open Trace Files Display windows.
Edit	Clear Trace File	Clears the trace file (to zero file size).
View	Toolbar	Shows or hides the toolbar.
	Status Bar	Shows or hides the status bar.
	Display Line Number	Shows or hides line numbers.
	Arrange Vertically	Arranges the open Trace Files Display windows top to tail on the desktop.
	Arrange Horizontally	Arranges the open Trace Files Display windows side by side on the desktop.
	Minimize All	Minimizes all the open Trace Files Display windows into icons.
	Refresh	Updates the contents of the displayed trace file.
Search	Find	Sets search criteria.
	Search Up	Searches for the search string to the beginning of the file.
	Search Down	Searches for the search string to the end of the file.
Help	Contents	Displays the contents of the JP1/Script online help.
	Search by Keyword	Lists the keywords of the JP1/Script online help.
	About JP1/Script	Displays version information for JP1/Script.

3.5.2 Mouse and key operations in the Script Trace Files Display window

(1) Mouse operations

Table 3-17 describes the mouse operation in the client area of the Script Trace Files Display window.

Table 3–17: Mouse operation in the Script Trace Files Display window

Operation	Processing
Click	Clears the previous selection and makes a new selection.

(2) Key operations

Table 3-18 describes the key operations in the client area of the Script Trace Files Display window.

Table 3–18: Key operations in the Script Trace Files Display window

Operation	Processing
Ctrl + B	Searches for text to the beginning of the trace file.
Ctrl + C	Copies selected text to the clipboard.
Ctrl + F	Displays the Search dialog box.
Ctrl + N	Searches for text to the end of the trace file.
Ctrl + P	Prints a trace file.
Del	Deletes a trace file.

3.6 Menu Editor operations

You can perform the following operations in the Script Menu Editor window:

- Create a new menu form
- Copy a menu form
- Paste a menu form
- Delete a menu form
- Change control attributes in one operation
- Align controls
- Space controls equally
- Center controls on a menu form
- Adjust controls to the same size
- Resize a control to text size
- Display a grid on a menu form
- Set the tab order
- Move a selected control to the foreground
- Move a selected control to the background
- Define the properties of a command
- Display a menu form in test view
- Print a menu form
- Create a new menu information file[#]
- Open an existing menu information file[#]
- Save a menu information file[#]
- Save a menu information file under a new name[#]
- Undo the previous edit[#]
- Redo the previous edit[#]
- Cut a selected control to the clipboard[#]
- Copy a selected control to the clipboard[#]
- Paste a control from the clipboard to a menu form[#]
- Select all controls in a menu form[#]
- Delete a selected control[#]
- Show or hide the toolbar[#]
- Show or hide the status bar[#]
- Show or hide control boxes[#]
- Show or hide a Properties dialog box[#]
- Show or hide the Command Properties dialog box[#]

- Display online help

#

These operations are not described. They are either standard Windows operations or simply involve choosing a command from a menu.

3.6.1 Script Menu Editor window and menus

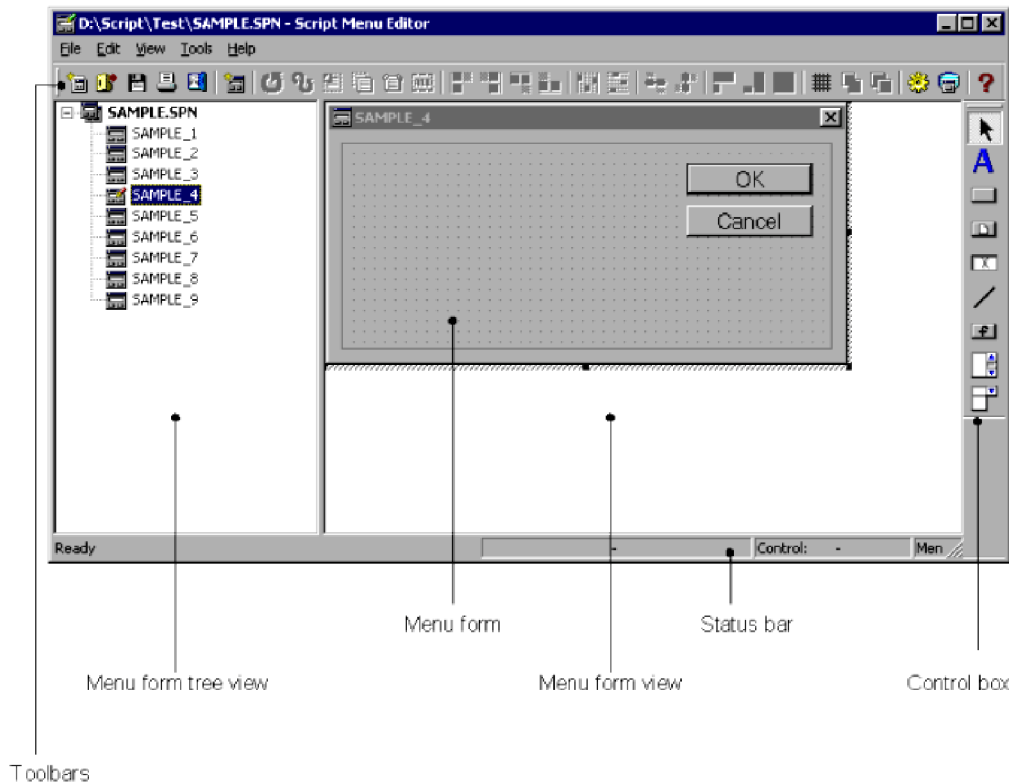
Menu Editor is for creating windows (menu forms) for completed script files. A menu form created with Menu Editor is saved as a menu information file. Use the `Menu` command to display the created window (menu form). For details about the `Menu` command, see *8.7.1 Menu (display a user-defined menu form)*.

This section describes the functions and menus of the Script Menu Editor window which appears when you start Menu Editor.

(1) Script Menu Editor window

Figure 3-6 shows the Script Menu Editor window and the names of its components.

Figure 3–6: Script Menu Editor window



(a) Toolbar

The toolbar contains buttons representing the most frequently used commands on the pull-down menus. By simply clicking a button, you can execute the corresponding command. You can hide the toolbar by toggling **Toolbar** in the **View** menu.

The following command buttons are displayed on the toolbar of the Script Menu Editor window.

Create

Creates a new menu information file.

Open

Opens a menu information file.

Save

Saves a menu information file.

Print Menu Form

Prints a menu form.

Exit

Quits Menu Editor.

Create Menu Form

Creates a new menu form.

Undo

Reverses the previous action.

Redo

Reinstates the previous action.

Cut

Cuts a selected control to the clipboard.

Copy

Copies a selected control to the clipboard.

Paste

Pastes a control from the clipboard to a menu form.

Select All

Selects all the controls written in a menu form.

Left Alignment

Aligns all controls along the left edge of a reference control.

Right Alignment

Aligns all controls along the right edge of a reference control.

Top Alignment

Aligns all controls along the top edge of a reference control.

Bottom Alignment

Aligns all controls along the bottom edge of a reference control.

Equal Horizontal Spacing

Arranges controls so that they are spaced equally in the horizontal direction.

Equal Vertical Spacing

Arranges controls so that they are spaced equally in the vertical direction.

Center Vertically

Aligns controls so that their vertical centers are in a horizontal line.

Center Horizontally

Aligns controls so that their horizontal centers are in a vertical line.

Adjust Width

Resizes a control to match the width of a selected control.

Adjust Height

Resizes a control to match the height of a selected control.

Adjust Size

Resizes a control to match the height and width of a selected control.

Set Grid

Sets a grid.

Move Forward

Moves a selected control to the foreground.

Move Backward

Moves a selected control to the background.

Set Command Properties

Defines the properties of a command.

Test View

Displays a created window (menu form) in test view.

Help

Displays the JP1/Script online help.

(b) Status bar

The status bar displays messages about the current processing being executed by Menu Editor and status messages at completion of processing.

(c) Menu form

A sheet on which you create a window. Information about the created window is written in a menu information file.

(d) Menu form tree view

A display area that shows in tree view a listing of the names of all menu forms written in menu information files. By clicking on a menu form, you can view its contents in the menu form view.

(e) Menu form view

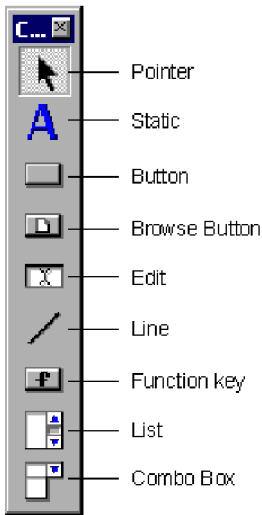
A display area that shows the contents of a menu information file.

(f) Control boxes

The control boxes that you can use on a menu form are displayed as buttons. To paste a control box, select a control box button, then drag and drop it onto the form.

Figure 3-7 shows the available control boxes.

Figure 3–7: Control boxes



Use the control boxes for the following purposes:

Pointer

To select a control.

Static

To paste a static (header) on a menu form.

Button

To paste a button on a menu form.

Browse Button

To paste a file browse button on a menu form.

Edit

To paste an edit on a menu form.

Line

To paste a line on a menu form.

Function Key

To paste a function key on a menu form. This control enables operation with a keyboard function key.

List

To paste a list on a menu form.

Combo Box

To paste a dialog box with an attached list box on a menu form.

(2) Menus in the Script Menu Editor window

The pull-down and pop-up menus in the Script Menu Editor window are described next.

(a) Commands in the pull-down menus

Table 3-19 lists the Menu Editor commands (functions) provided in the pull-down menus. Table 3-20 lists the cascading menu and commands that open from the **Layout** command.

Table 3–19: Menu Editor menus and commands

Menu	Command (function)	Description
File	Create	Creates a new menu information file.
	Open	Opens a menu information file.
	Save	Saves a menu information file.
	Save As	Saves a selected menu information file under a new name.
	Print Menu Form	Prints a menu form.
	Exit	Quits Menu Editor.
	<i>menu-information-file-name</i>	Displays the name of the most recently saved menu information file.
Edit	Create Menu Form	Creates a new menu form.
	Copy Menu Form	Copies an existing menu form to the clipboard.
	Paste Menu Form	Pastes a menu form from the clipboard.
	Delete Menu Form	Deletes a menu form from the clipboard.
	Undo	Reverses the previous action.
	Redo	Reinstates the previous action.
	Cut	Cuts a selected control to the clipboard.
	Copy	Copies a selected control to the clipboard.
	Paste	Pastes a control from the clipboard to a menu form.
	Select All	Selects all controls written on a menu form.
	Delete	Deletes a selected control.
	Change as Batch	Changes the attributes of selected controls in one operation.
	Layout	Specifies the layout of the controls pasted on a menu form. A cascading menu is attached to the Layout command (see Table 3-20).
View	Toolbar	Shows or hides the toolbar.
	Status Bar	Shows or hides the status bar.
	Control Box	Shows or hides the control boxes.
	Properties	Shows or hides a Properties dialog box.
	Command Properties	Shows or hides the Command Properties dialog box.
Tools	Set Command Properties	Defines the properties of a command.
	Test View	Displays a created window (menu form) in test view.
Help	Contents	Displays the contents of the JP1/Script online help.
	Search by Keyword	Lists the keywords of the JP1/Script online help.
	About JP1/Script	Displays version information for JP1/Script.

Table 3–20: Cascading menu of the Layout command

Cascading menu	Command (function)	Description
Align	Left	Aligns controls along the left edge of a reference control.
	Right	Aligns controls along the right edge of a reference control.
	Top	Aligns controls along the top edge of a reference control.
	Bottom	Aligns controls along the bottom edge of a reference control.
	Center Vertically	Aligns controls so that their vertical centers are in a horizontal line with a reference control.
	Center Horizontally	Aligns controls so that their horizontal centers are in a vertical line with a reference control.
Equal Spacing	Horizontal	Arranges controls so that they are spaced equally in the horizontal direction.
	Vertical	Arranges controls so that they are spaced equally in the vertical direction.
Arrange on Menu Form	Center Vertically	Centers a control vertically on the menu form.
	Center Horizontally	Centers a control horizontally on the menu form.
Set Same Size	Width	Resizes a control to match the width of a selected control.
	Height	Resizes a control to match the height of a selected control.
	Width and Height	Resizes a control to match the height and width of a selected control.
Adjust Size to Text	--	Resizes a control to fit the text size.
Set Grid	--	Sets a grid.
Set Tab Order	--	Sets the tab order for moving among controls.
Move Forward	--	Moves a selected control to the foreground.
Move Backward	--	Moves a selected control to the background.

(b) Pop-up menu

To display the pop-up menu in the Script Menu Editor window, right-click in the menu form view. Table 3-21 lists the commands in the displayed pop-up menu.

Table 3–21: Commands in the Script Menu Editor pop-up menu

Command (function)	Description
Create	Creates a new menu information file.
Open	Opens an existing menu information file.
Save	Saves a menu information file, replacing its previous contents.
Print Menu Form	Prints a menu form.
Copy Menu Form	Copies an existing menu form to the clipboard.
Paste Menu Form	Pastes a menu form from the clipboard.
Delete Menu Form	Deletes an existing menu form.
Undo	Reverses the previous action.

Command (function)	Description
Redo	Reinstates the previous action.
Cut	Cuts a selected control to the clipboard.
Copy	Copies a selected control to the clipboard.
Paste	Pastes a control from the clipboard to a menu form.
Select All	Selects all controls written on a menu form.
Delete	Deletes a selected control.
Change as Batch	Changes the attributes of selected controls in one operation.
Layout	Specifies the layout of the controls pasted on a menu form. A cascading menu is attached to the Layout command (see Table 3-20).
Set Command Properties	Defines the properties of a command.
Test View	Displays a created window (menu form) in test view.
Properties	Displays a Properties dialog box for a selected control.

3.6.2 Mouse and key operations in the Script Menu Editor window

(1) Mouse operations

Table 3-22 lists mouse operations in the client area of the Script Menu Editor window.

Table 3–22: Mouse operations in the Script Menu Editor window

Operation	Processing
Click	Makes a selection or clears a previous selection.
Double-click	Displays a Properties dialog box for the menu form or for a control.
Right-click	Displays a pop-up menu.
Drag	Drags or selects a control. When you drag across multiple controls, the selection order is according to their YX positions, starting from top left and ending at bottom right.
Ctrl + drag	Copies a selected control.

(2) Key operations

Table 3-23 lists key operations in the client area of the Script Menu Editor window and shows whether each operation is supported in edit mode and monitoring mode.

Table 3–23: Key operations in the Script Menu Editor window

Operation	Processing
Alt + 1	Shows or hides a control box.
Alt + 2	Shows or hides a Properties dialog box.
Alt + 3	Shows or hides the Command Properties dialog box.
Alt + F4	Quits Menu Editor.

Operation	Processing
Alt + R	Arranges all controls so that they are horizontally equidistant from one another.
Alt + D	Arranges all controls so that they are vertically equidistant from one another.
Ctrl + A	Selects all controls written on a menu form.
Ctrl + C	Copies a selected control.
Ctrl + F	Creates a new menu form.
Ctrl + K	Defines the properties of a command.
Ctrl + N	Creates a new menu information file.
Ctrl + O	Opens a menu information file.
Ctrl + P	Prints a menu form.
Ctrl + S	Saves a menu information file.
Ctrl + V	Pastes the clipboard contents to a menu form.
Ctrl + X	Cuts a selected control.
Ctrl + Z	Undoes the previous edit.
Ctrl + F9	Centers controls vertically in the dialog box.
Ctrl + L	Aligns controls along the left edge of a reference control.
Ctrl + R	Aligns controls along the right edge of a reference control.
Ctrl + U	Aligns controls along the top edge of a reference control.
Ctrl + D	Aligns controls along the bottom edge of a reference control.
Delete	Deletes a selected menu form or control.
F5	Displays a menu form in test view.
F7	Resizes a control to fit the text size.
F9	Aligns controls so that their vertical centers are in a horizontal line with a reference control.
Shift + F9	Aligns controls so that their horizontal centers are in a vertical line with a reference control.
Shift + Ctrl + Z	Redoes the previous edit.
Shift + Ctrl + F9	Centers controls horizontally in the dialog box.
←	Moves one or more controls to the left (between gridlines).
→	Moves one or more controls to the right (between gridlines).
↑	Moves one or more controls upward (between gridlines).
↓	Moves one or more controls downward (between gridlines).
Alt + ←	Moves one or more controls to the left (in pixel units).
Alt + →	Moves one or more controls to the right (in pixel units).
Alt + ↑	Moves one or more controls upward (in pixel units).
Alt + ↓	Moves one or more controls downward (in pixel units).
Shift + ←	Resizes one or more selected controls leftward from the top left corner (between gridlines). [#]
Shift + →	Resizes one or more selected controls rightward from the top left corner (between gridlines). [#]

Operation	Processing
Shift + ↑	Resizes one or more selected controls upward from the top left corner (between gridlines). [#]
Shift + ↓	Resizes one or more selected controls downward from the top left corner (between gridlines). [#]
Shift + Alt + ↑	Resizes one or more selected controls leftward from the top left corner (in pixel units). [#]
Shift + Alt + ↓	Resizes one or more selected controls rightward from the top left corner (in pixel units). [#]
Shift + Alt + ←	Resizes one or more selected controls upward from the top left corner (in pixel units). [#]
Shift + Alt + →	Resizes one or more selected controls downward from the top left corner (in pixel units). [#]
Enter	Displays the properties of a selected control.

[#]: This operation is invalid when the control is a line.

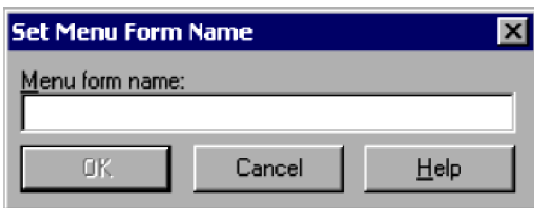
3.6.3 Creating a menu form

The following describes how to create a menu form using Script Menu Editor. A *menu form* is a window displayed on the screen when the user executes a created script file. On a menu form, you can arrange buttons, list boxes, and other graphics. These are called controls.

To create a menu form:

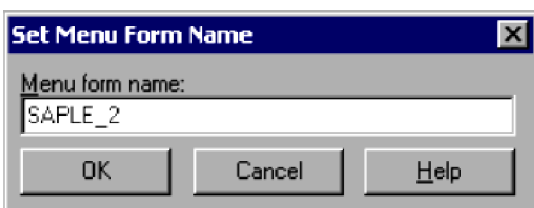
1. Choose **Edit, Create Menu Form**.

The Set Menu Form Name dialog box appears.

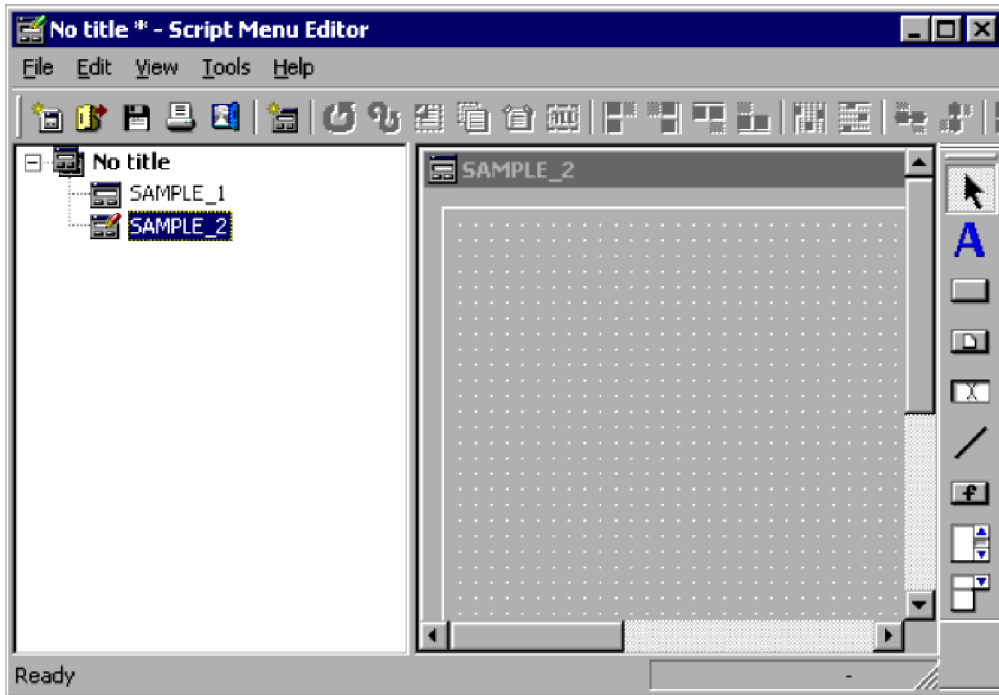


For details about using this dialog box, see [4.4.38 Set Menu Form Name dialog box](#).

2. Enter the menu form name.



3. Click the **OK** button.

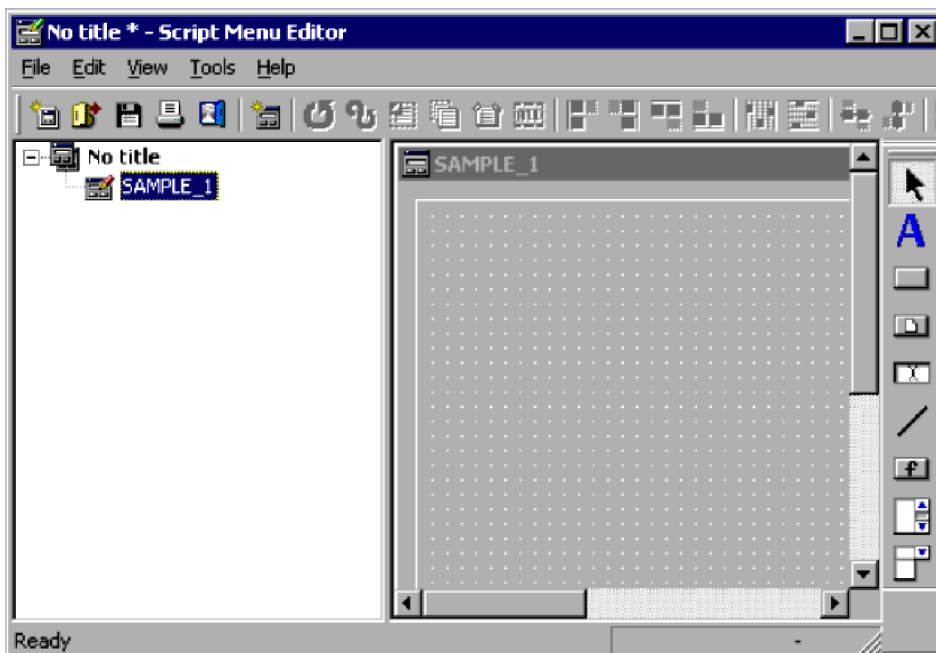


A new menu form appears in the menu form view area, and the name you entered at step 2 adds in the menu form tree view.

3.6.4 Copying a menu form

To copy an existing menu form:

1. In the menu form tree view, select a menu form to copy.



2. Choose **Edit, Copy Menu Form**.

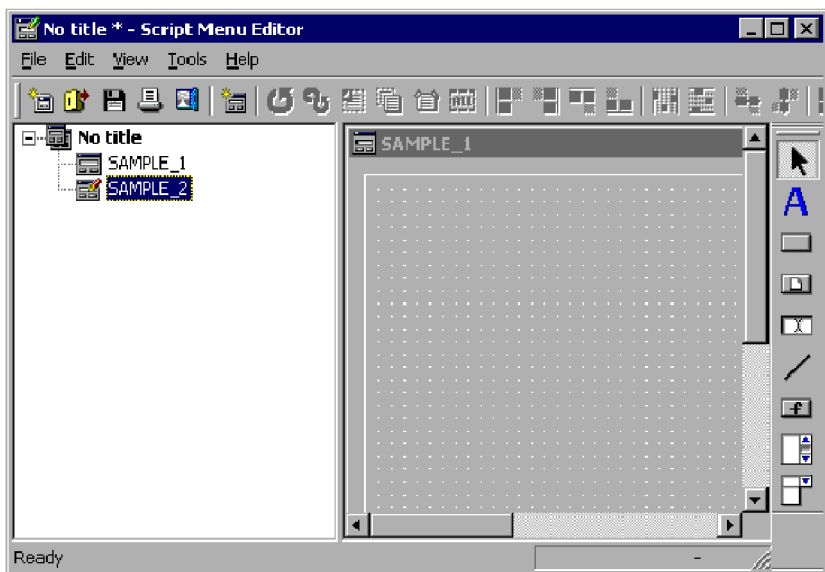
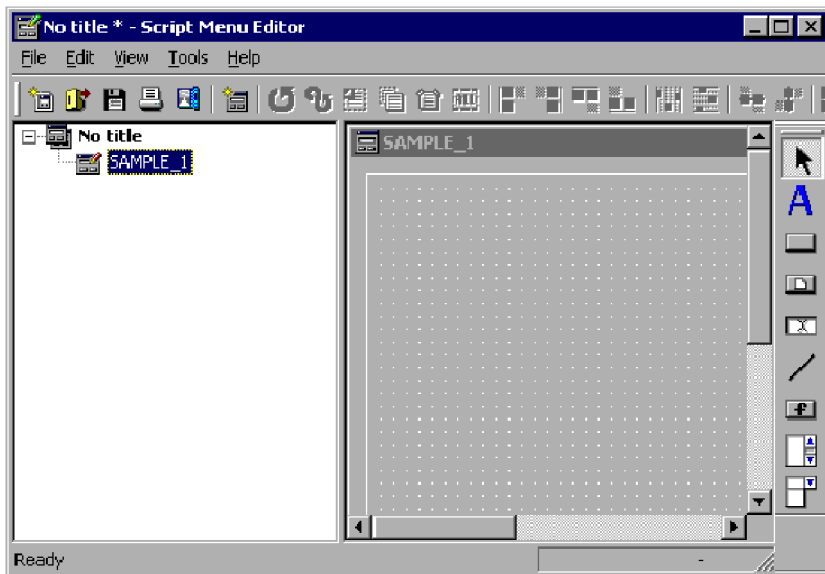
The selected menu form is copied to the clipboard.

3.6.5 Pasting a menu form

To paste a menu form from the clipboard:

1. Choose **Edit, Paste Menu Form**.

The menu form is pasted from the clipboard to the menu form view, and its name appears in the menu form tree view.

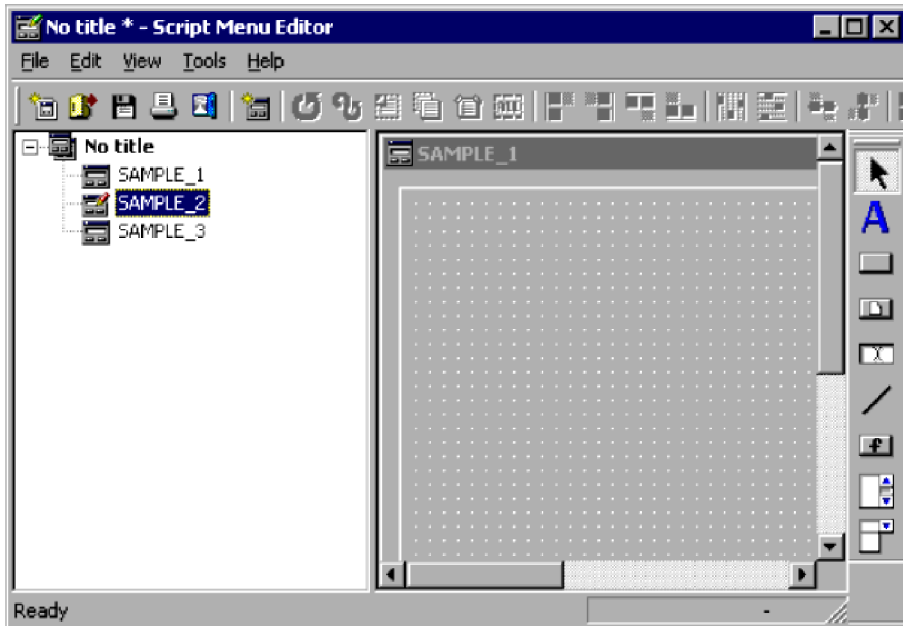


If a menu form of the same name already exists, a sequential number is added to the name, starting from 1 (example: *name* → *name1*, *name1* → *name2*).

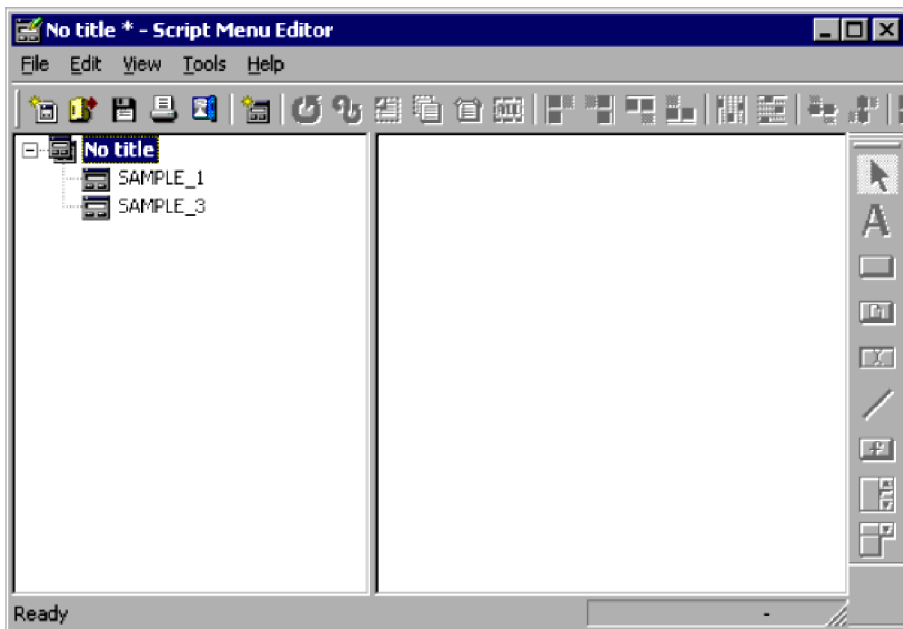
3.6.6 Deleting a menu form

To delete a completed menu form:

1. Select a menu form in the menu form tree view.



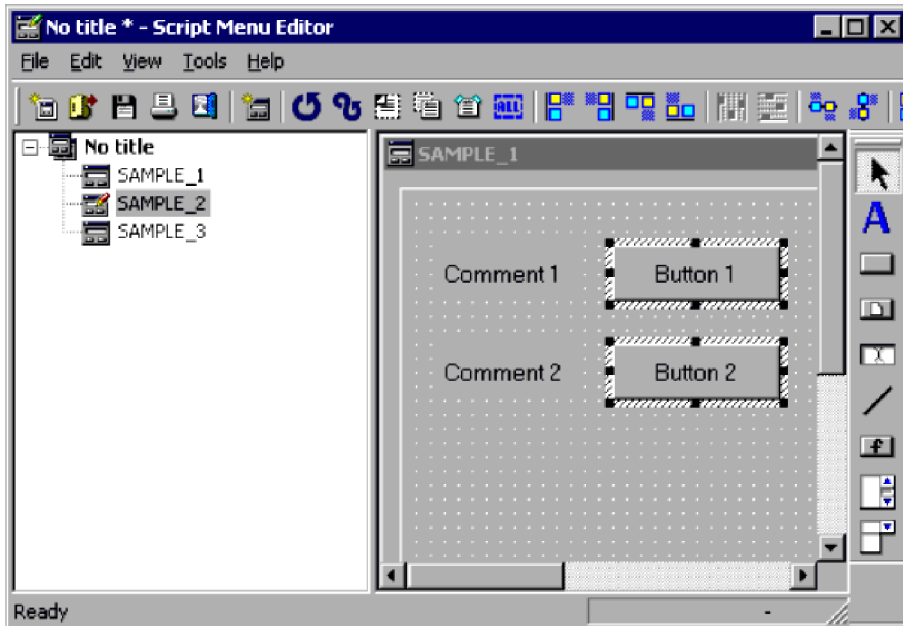
2. Choose **Edit, Delete Menu Form**.
The selected menu form is deleted.



3.6.7 Changing control attributes in one operation

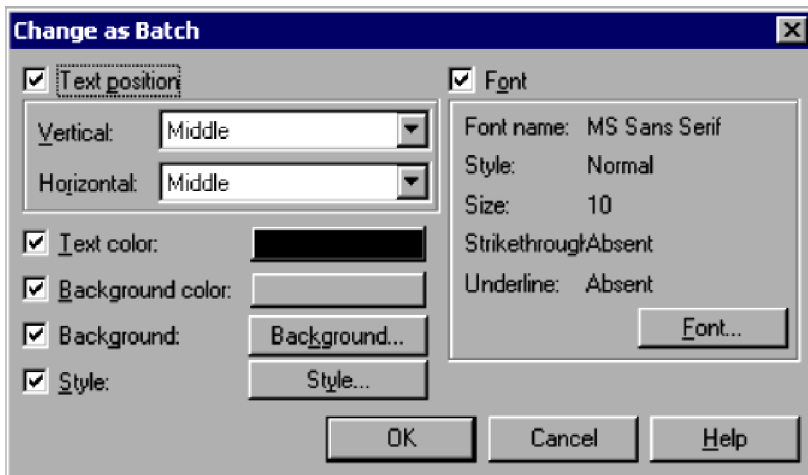
To change the attributes of controls in a menu form in one operation:

1. Select the controls you want to change in the menu form.
In this example, buttons 1 and 2 are selected.



2. Choose **Edit, Change as Batch**.

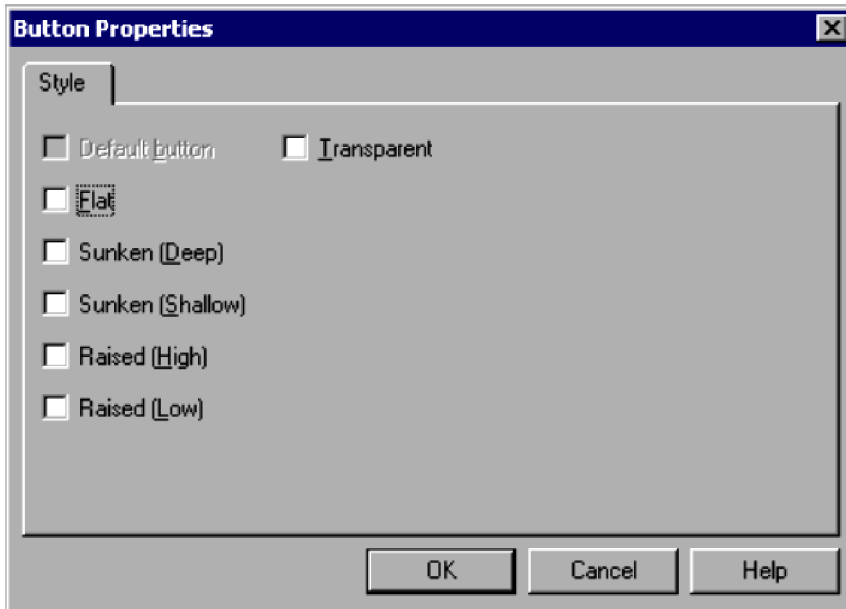
The Change as Batch dialog box appears.



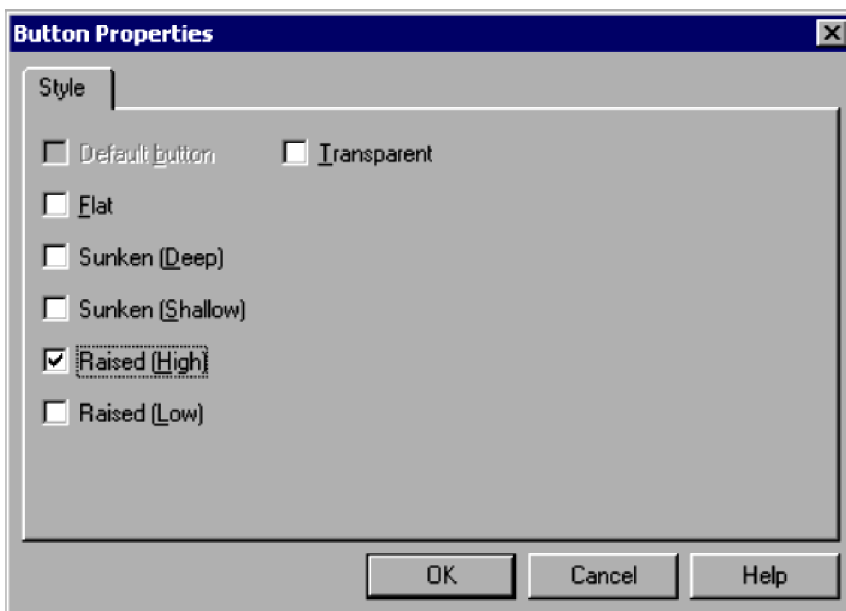
For details about using this dialog box, see [4.4.39 Change as Batch dialog box](#).

3. Enter the new settings.

In this example, click the **Style** button to change the style.

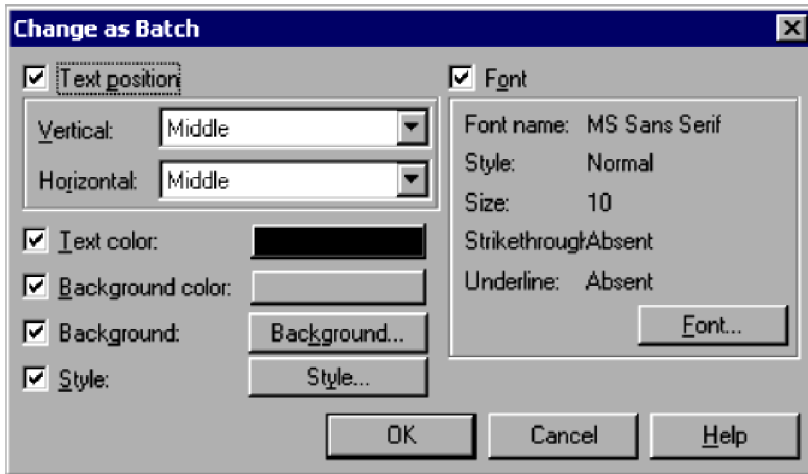


4. In the Button Properties dialog box, select **Raised (High)**.



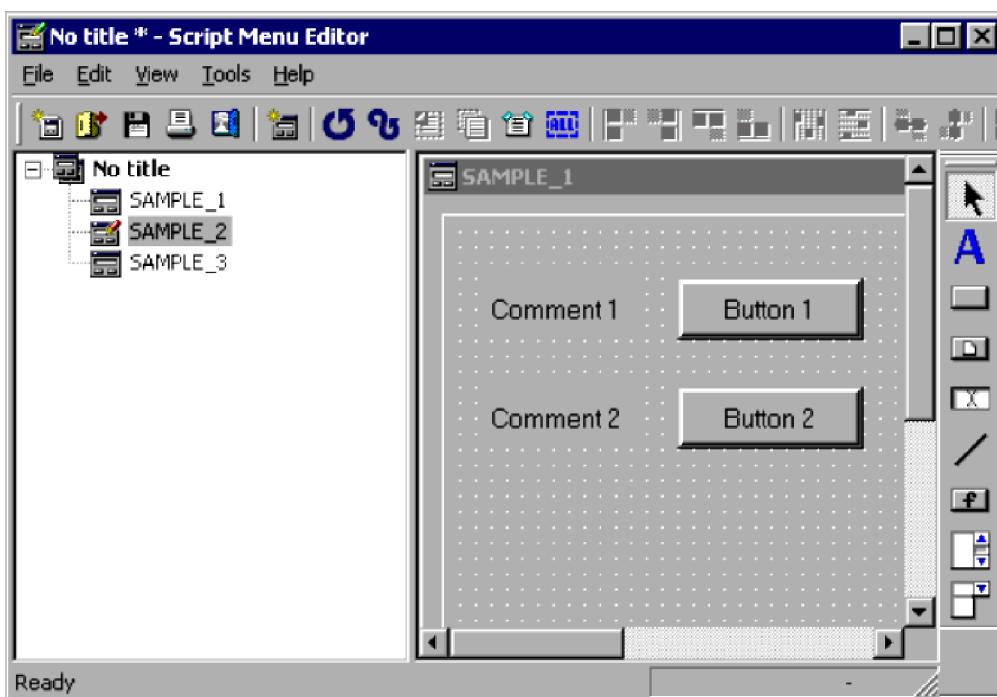
5. Click **OK**.

The Change as Batch dialog box appears again.



6. Click **OK**.

The controls are set with the new setting you entered at step 3.



Notes

- The **Change as Batch** command is disabled when no controls are selected.
- The **Change as Batch** command is disabled if the selected controls include a line.

3.6.8 Aligning controls

You can arrange two or more controls in a row. The position of the control you select first is taken as the reference point, and the other controls are moved in relation to that position.

To align controls:

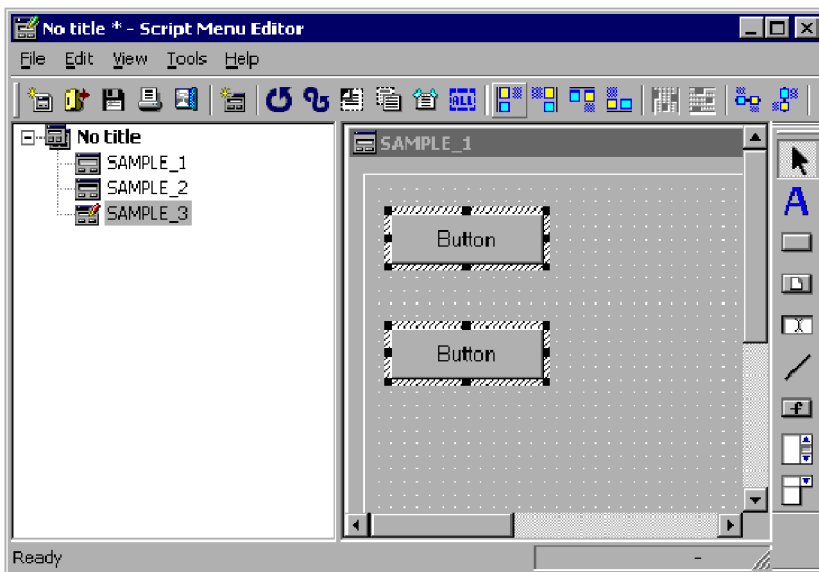
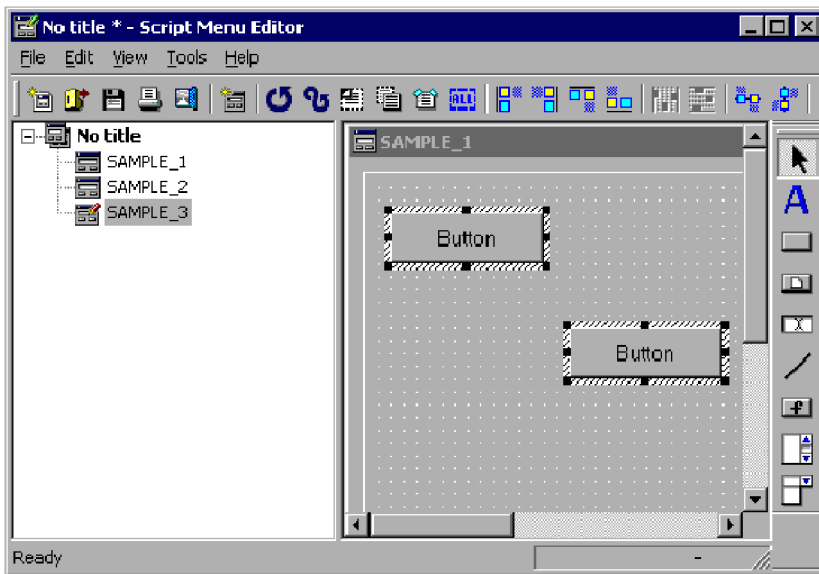
1. Select two or more controls on the menu form.

2. Arrange the second and subsequent controls in relation to the control you selected first.

This operation depends on how you want to arrange the controls.

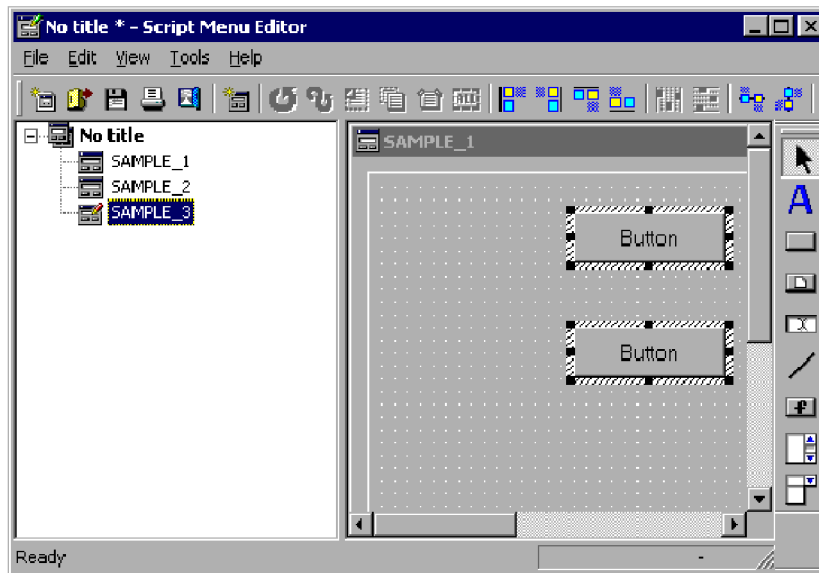
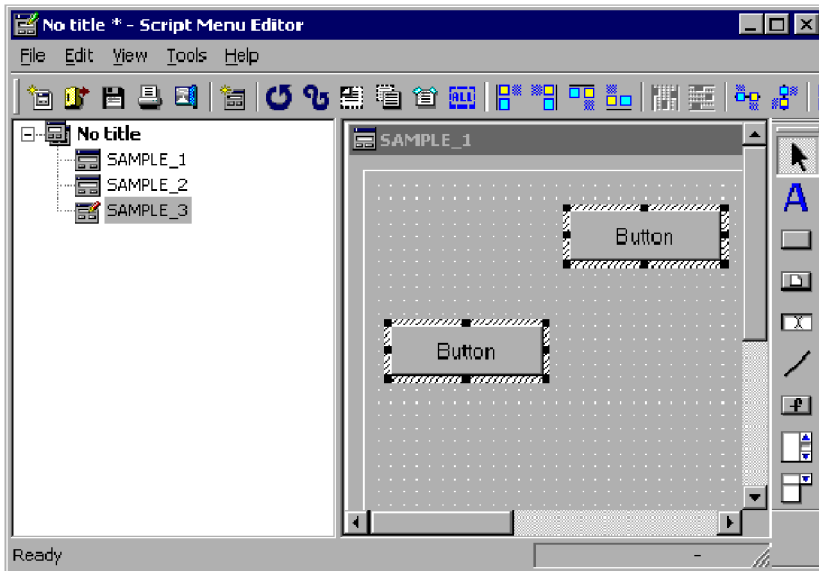
To align the controls at their left edges:

Choose **Edit, Layout, Align, Left**.

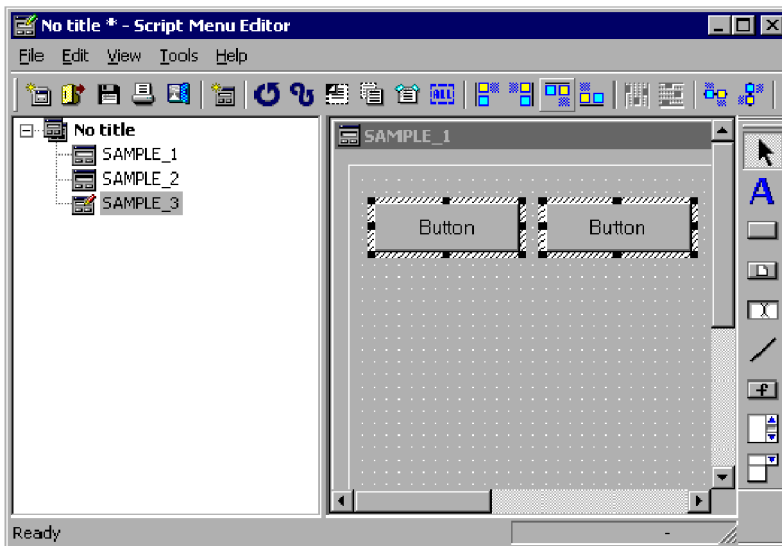
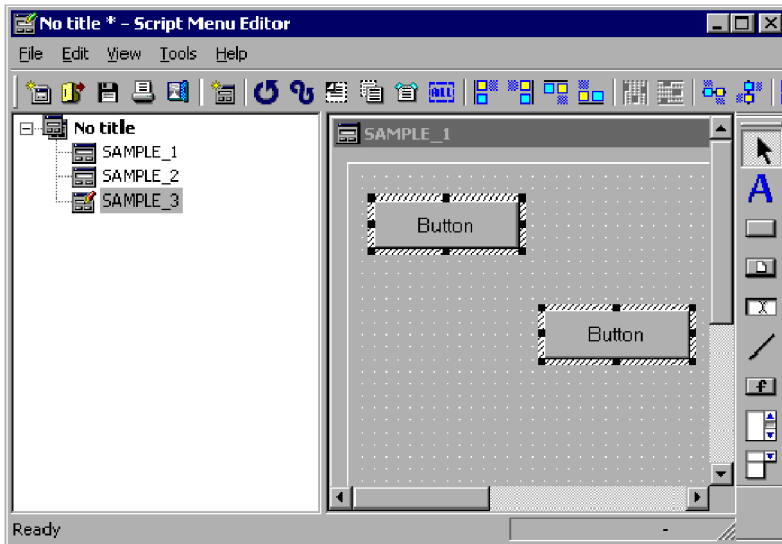


To align the controls at their right edges:

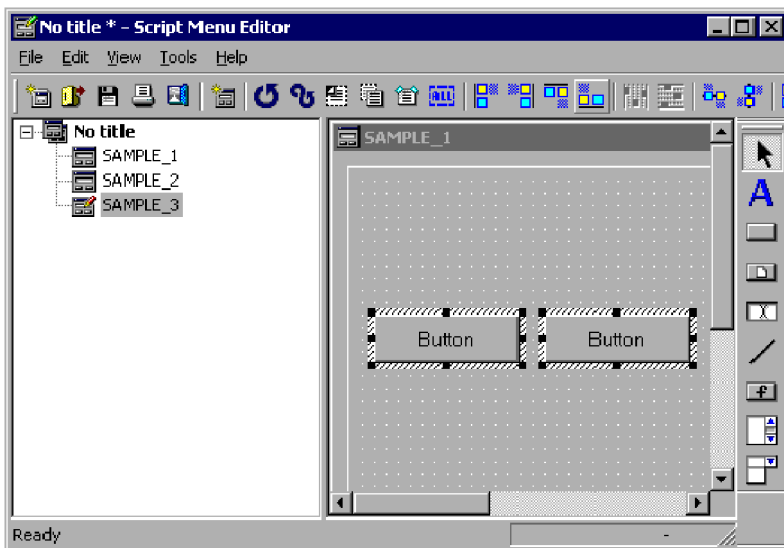
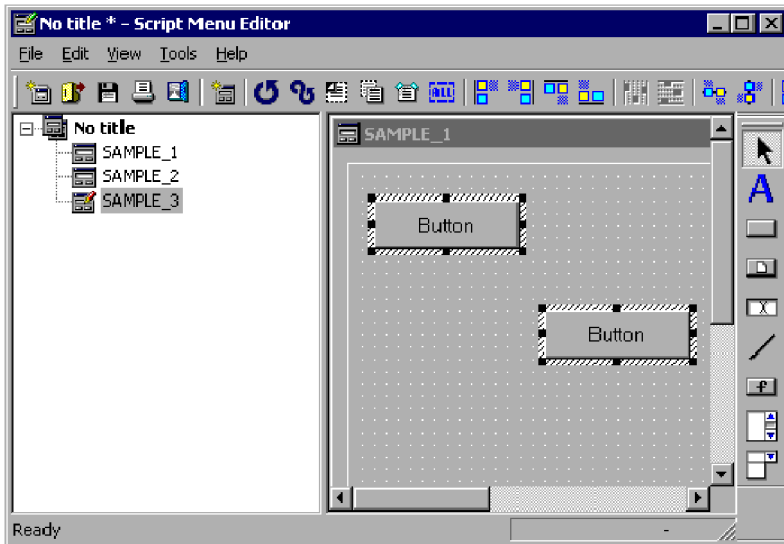
Choose **Edit, Layout, Align, Right**.



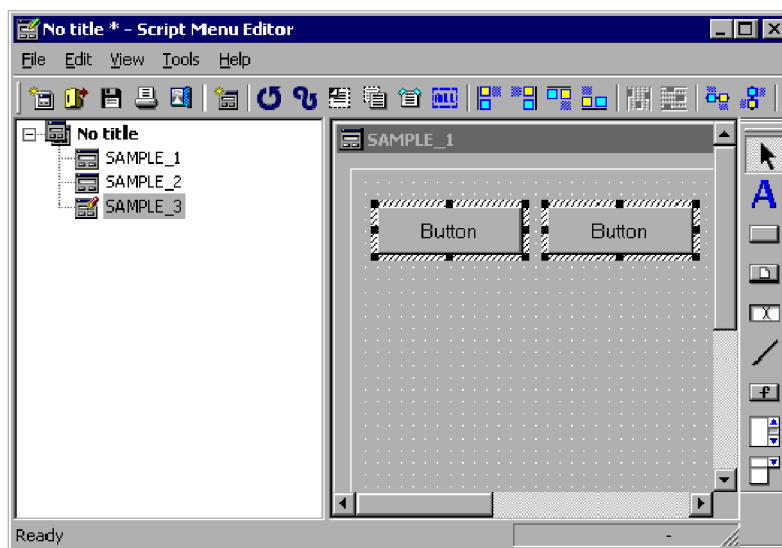
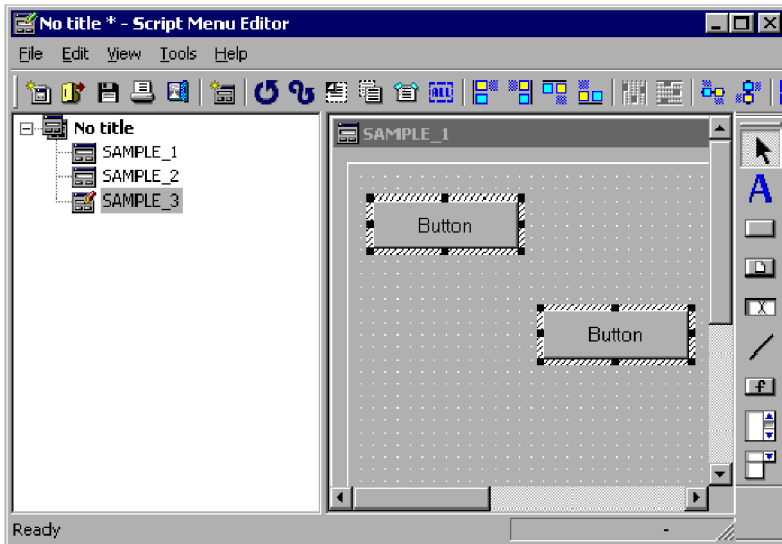
To align the controls at their top edges:
Choose **Edit, Layout, Align, Top**.



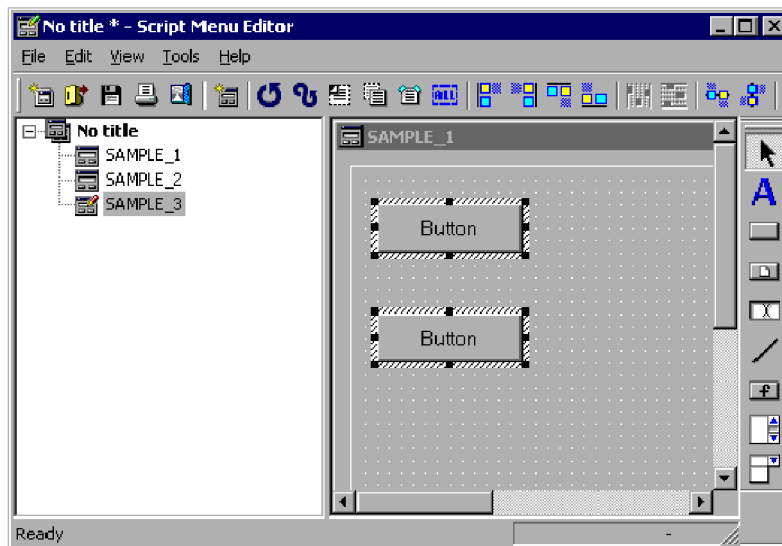
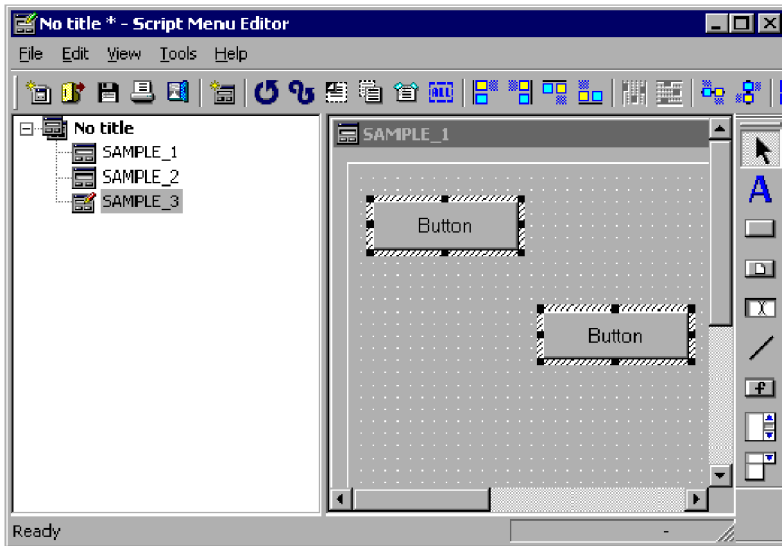
To align the controls at their bottom edges:
Choose **Edit, Layout, Align, Bottom**.



**To align the controls so that their vertical centers are in a horizontal line:
Choose Edit, Layout, Align, Center Vertically.**



**To align the controls so that their horizontal centers are in a vertical line:
Choose Edit, Layout, Align, Center Horizontally.**



The selected controls are aligned in the specified direction.

Note

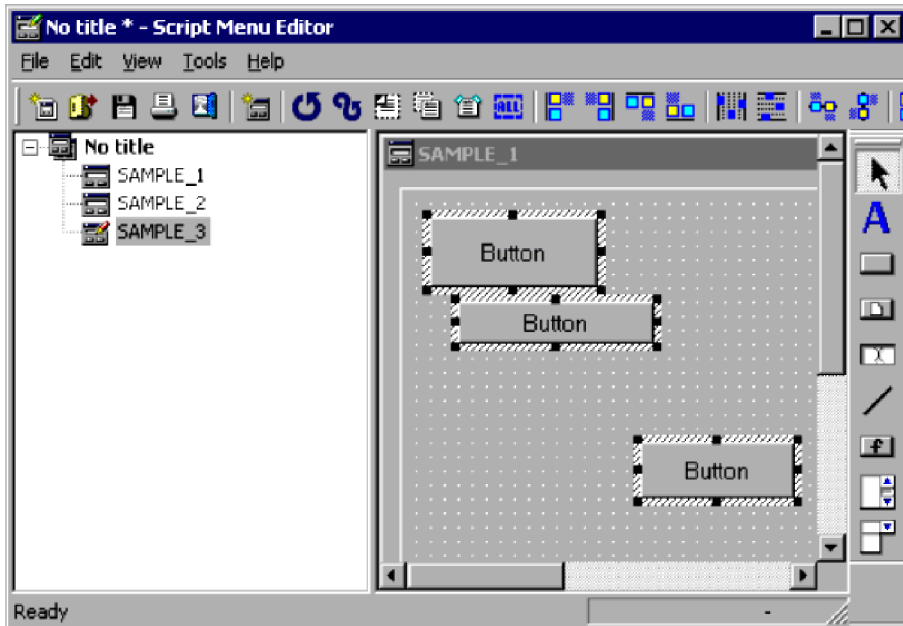
The **Align** command is disabled unless two or more controls are selected.

3.6.9 Spacing controls equally

You can arrange three or more controls so that they are evenly spaced in the vertical or horizontal direction.

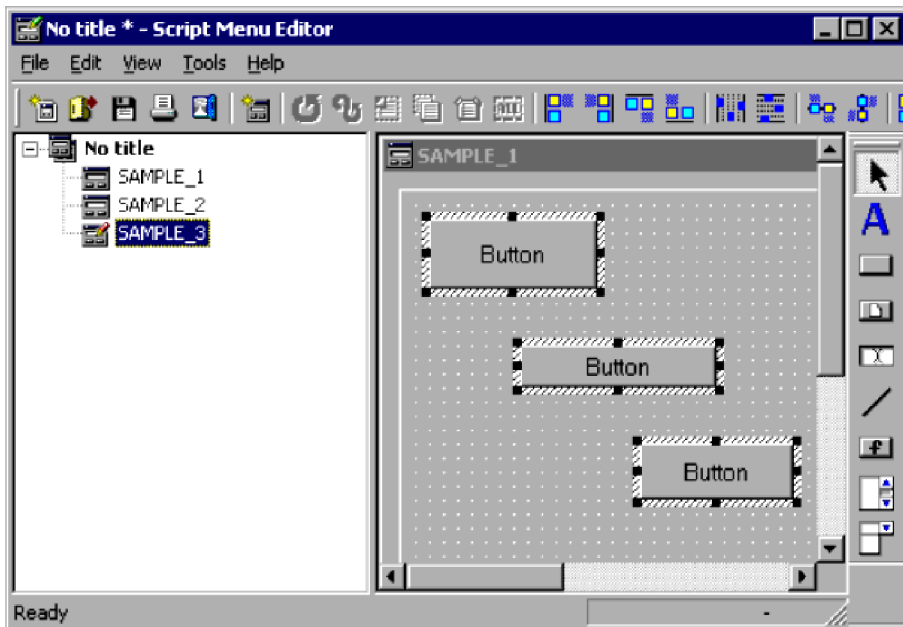
To arrange controls with equal spacing in the horizontal direction:

1. Select three or more controls in the menu form.



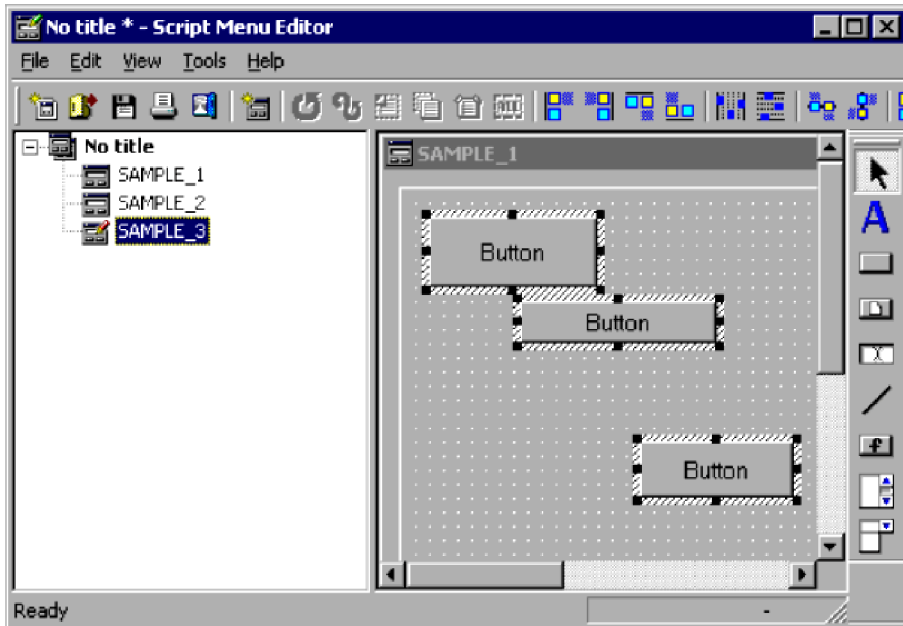
2. Choose **Edit, Layout, Equal Spacing, Horizontal**.

The control(s) in the middle are evenly spaced between the leftmost and rightmost of the selected controls.



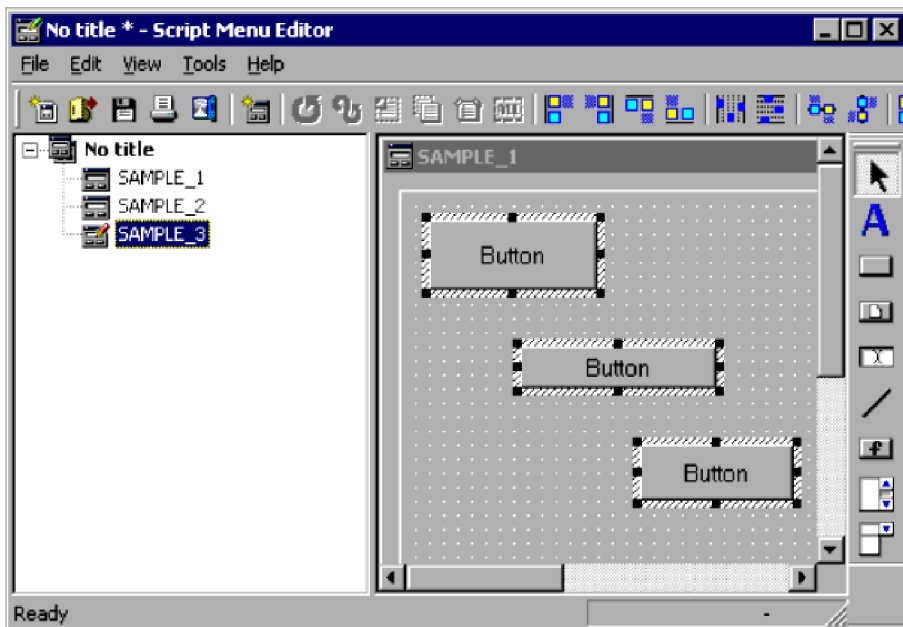
To arrange controls with equal spacing in the vertical direction:

1. Select three or more controls in the menu form.



2. Choose **Edit, Layout, Equal Spacing, Vertical**.

The control(s) in the middle are evenly spaced between the topmost and bottommost of the selected controls.



Note

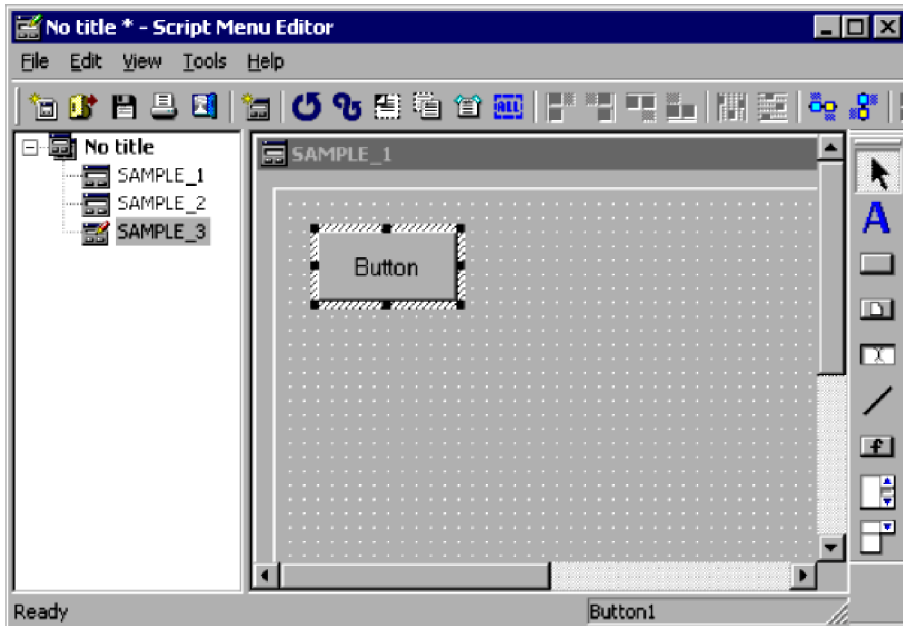
The **Equal Spacing** command is disabled unless three or more controls are selected.

3.6.10 Centering controls on a menu form

You can center a control in the vertical or horizontal direction on a menu form.

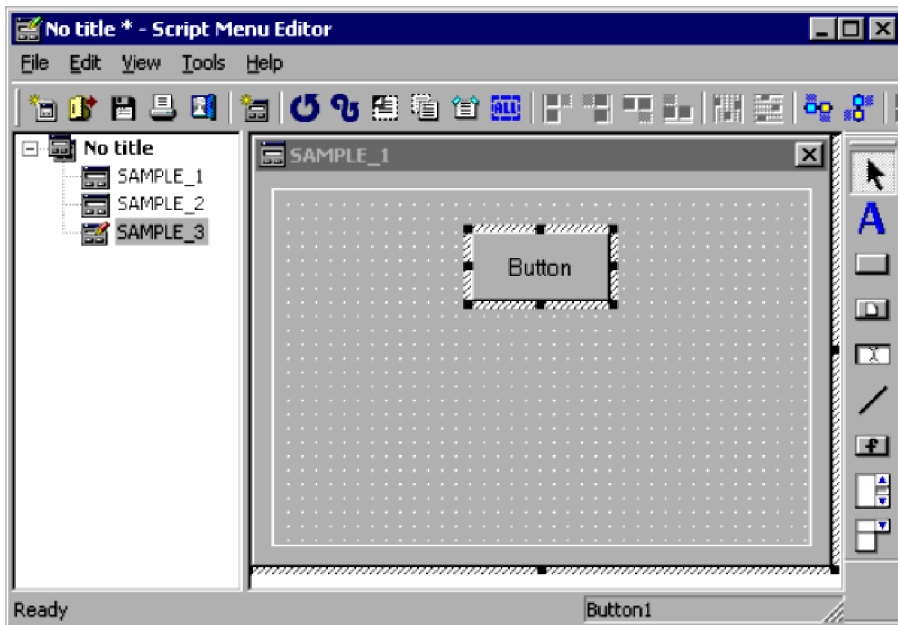
To center a control in the horizontal direction:

1. Select a control in the menu form.



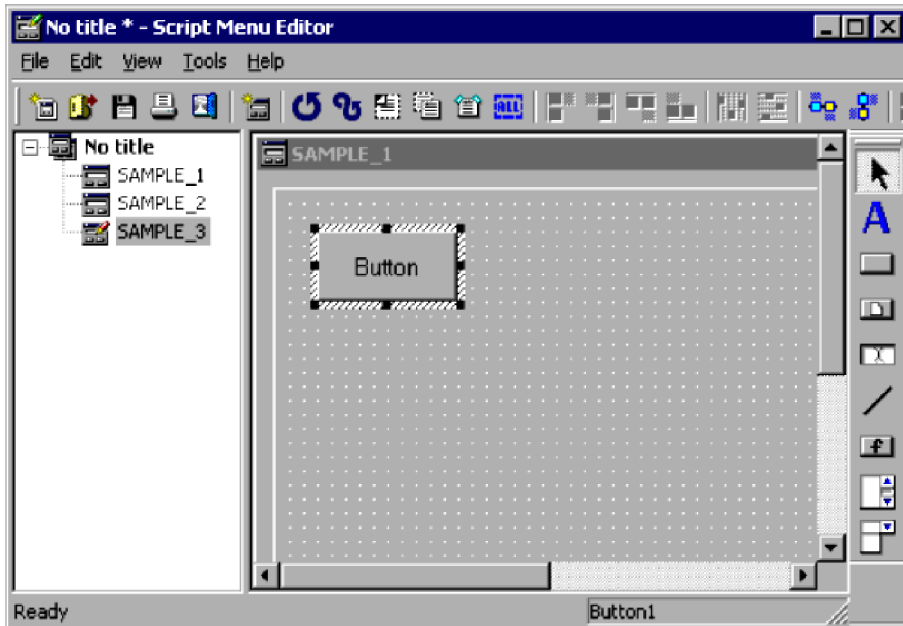
2. Choose **Edit, Layout, Arrange on Menu Form, Center Horizontally.**

The selected control is centered in the horizontal direction on the menu form.



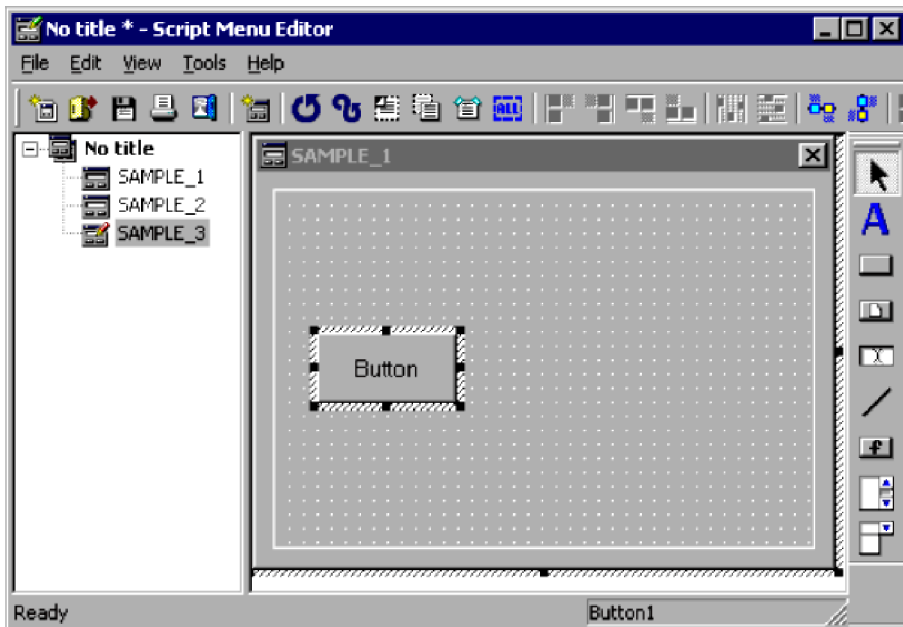
To center a control in the vertical direction:

1. Select a control in the menu form.



2. Choose **Edit, Layout, Arrange on Menu Form, Center Vertically**.

The selected control is centered in the vertical direction on the menu form.

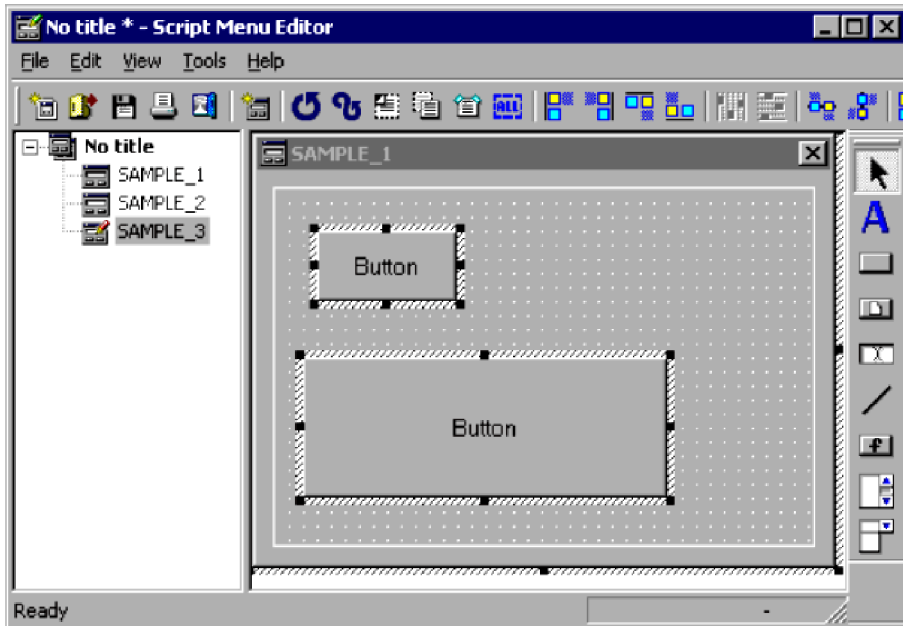


3.6.11 Adjusting controls to the same size

You can resize one or more controls to match a selected control. You can match the width, height, or both.

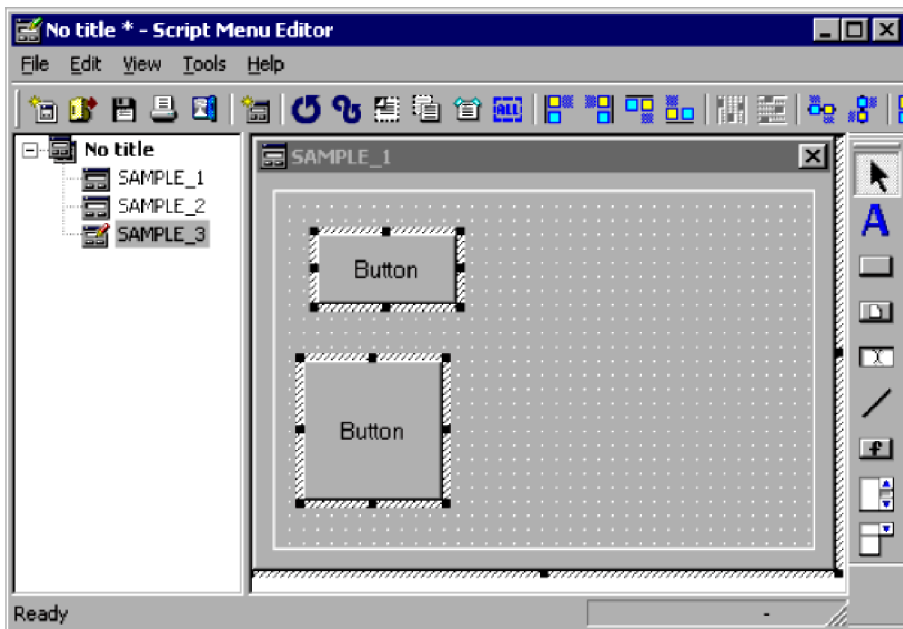
To match control width:

1. Select two or more controls in the menu form.



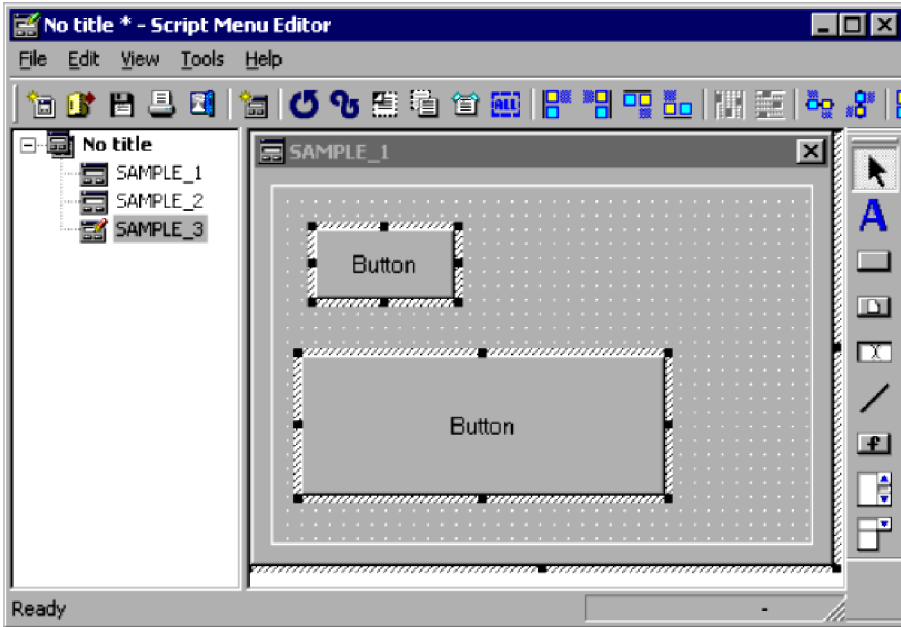
2. Choose **Edit, Layout, Set Same Size, Width.**

The second and subsequent controls are resized to the width of the control you selected first.



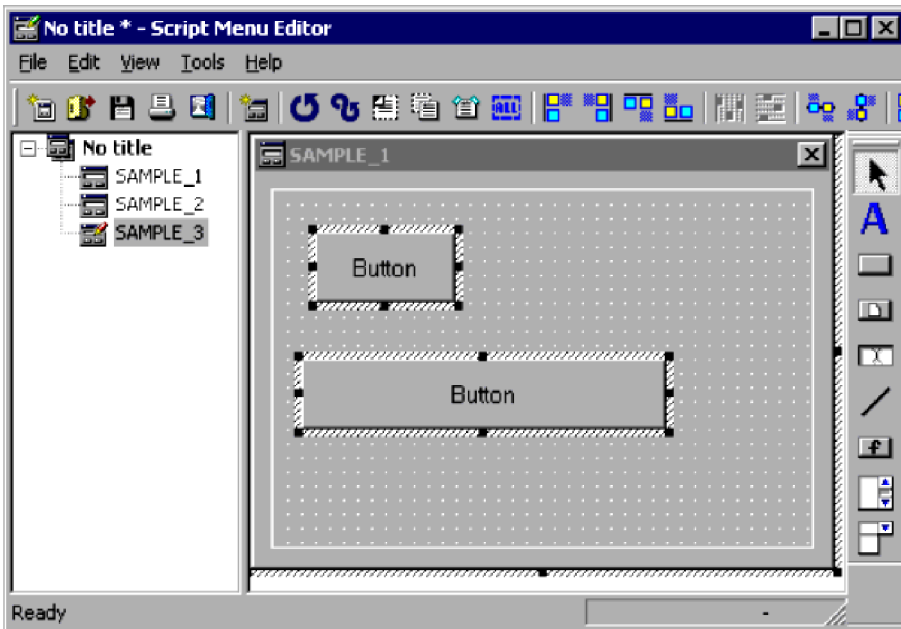
To match control height:

1. Select two or more controls in the menu form.



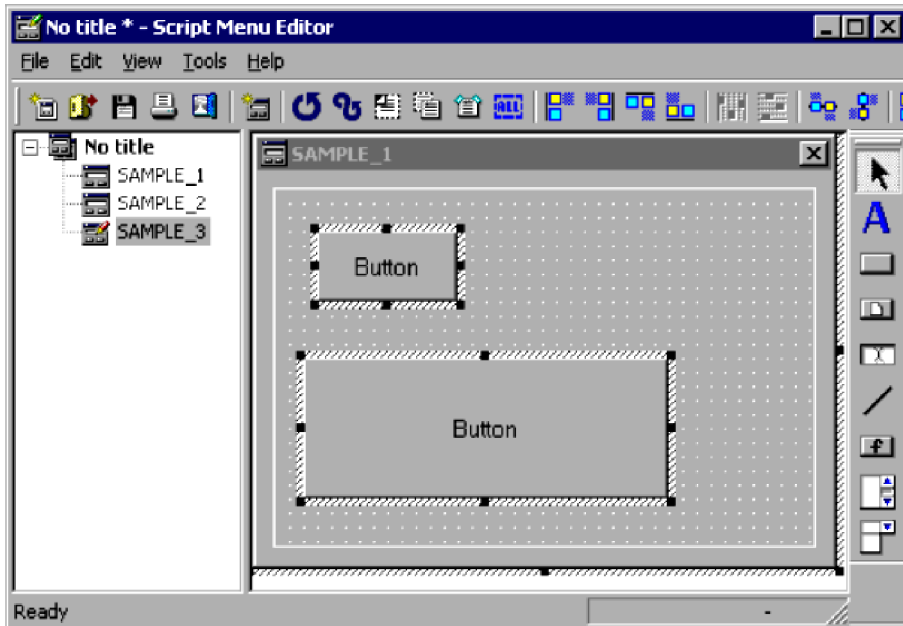
2. Choose **Edit, Layout, Set Same Size, Height**.

The second and subsequent controls are resized to the height of the control you selected first.



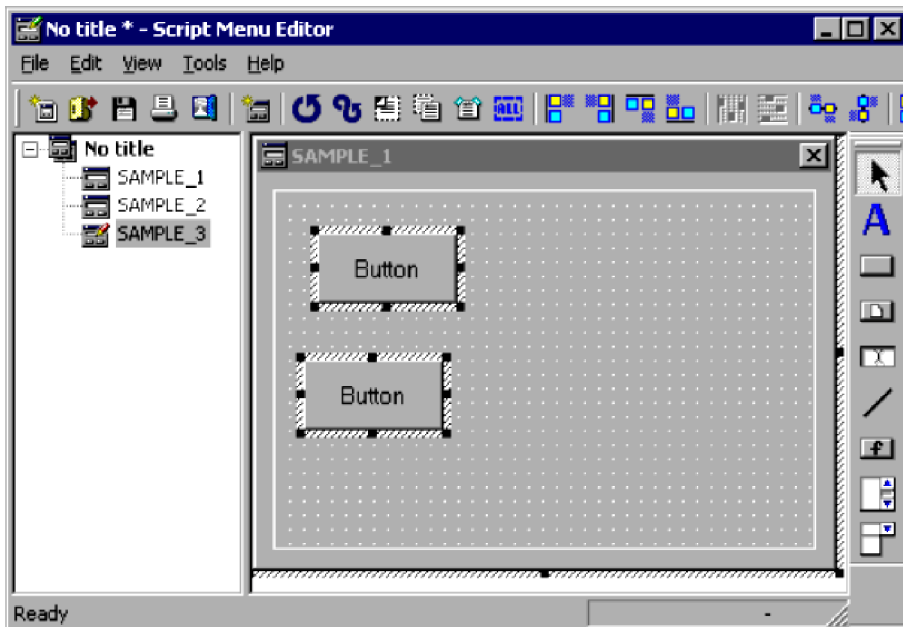
To match control width and height:

1. Select two or more controls in the menu form.



2. Choose **Edit, Layout, Set Same Size, Width and Height.**

The second and subsequent controls are resized to the width and height of the control you selected first.



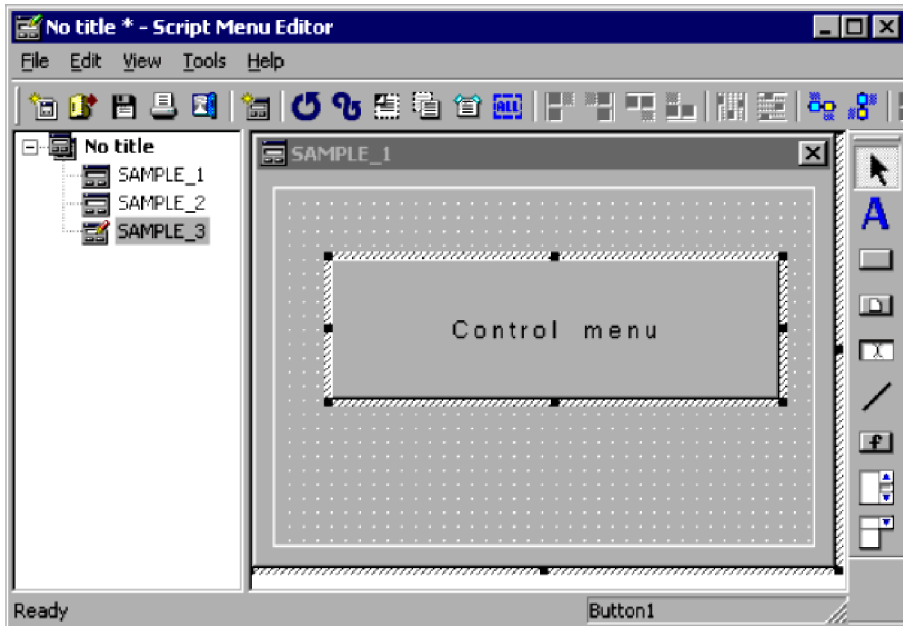
Note

The **Set Same Size** command is disabled unless two or more controls are selected.

3.6.12 Resizing a control to text size

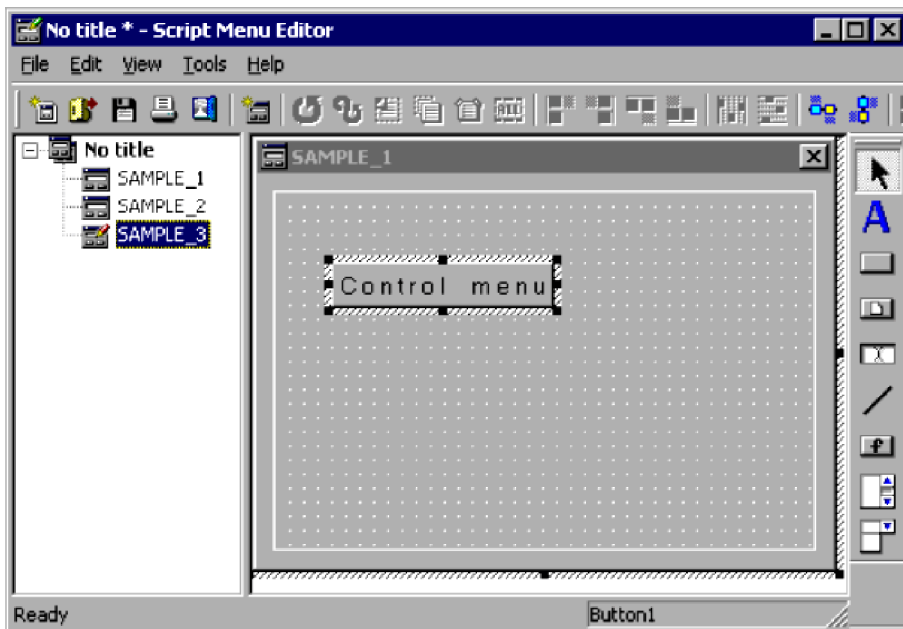
To resize a selected control to fit the text size:

1. Select a control in the menu form.



2. Choose **Edit, Layout, Adjust Size To Text**.

The selected control is resized to fit the text size.



Notes

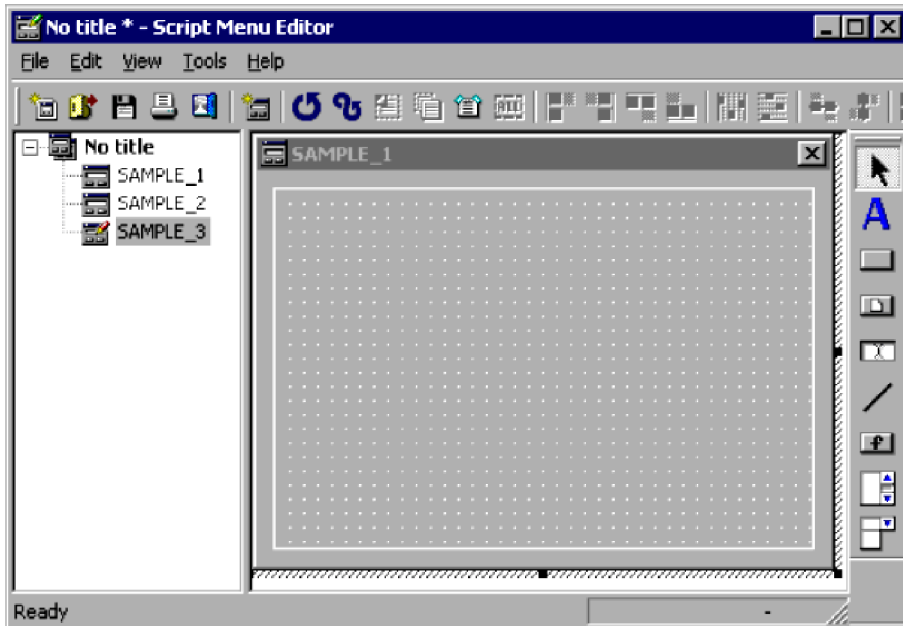
- The **Adjust Size To Text** command is disabled when no control is selected.
- The **Adjust Size To Text** command is disabled if the selected control is not a static, button, or browse button.

3.6.13 Displaying a grid on a menu form

To display a grid on a menu form and set the grid spacing:

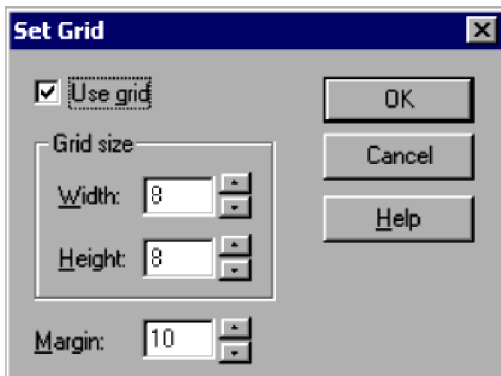
1. Choose **Edit, Layout, Set Grid**.

The Set Grid dialog box appears.



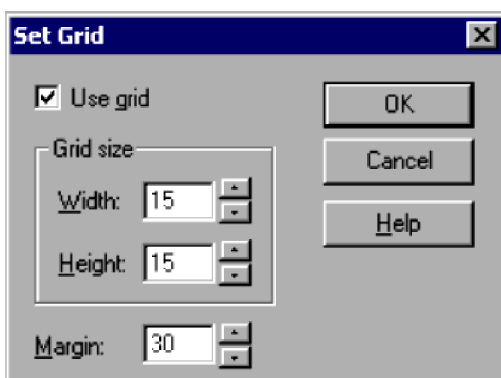
For details about using this dialog box, see [4.4.40 Set Grid dialog box](#).

2. Select the **Use grid** check box.



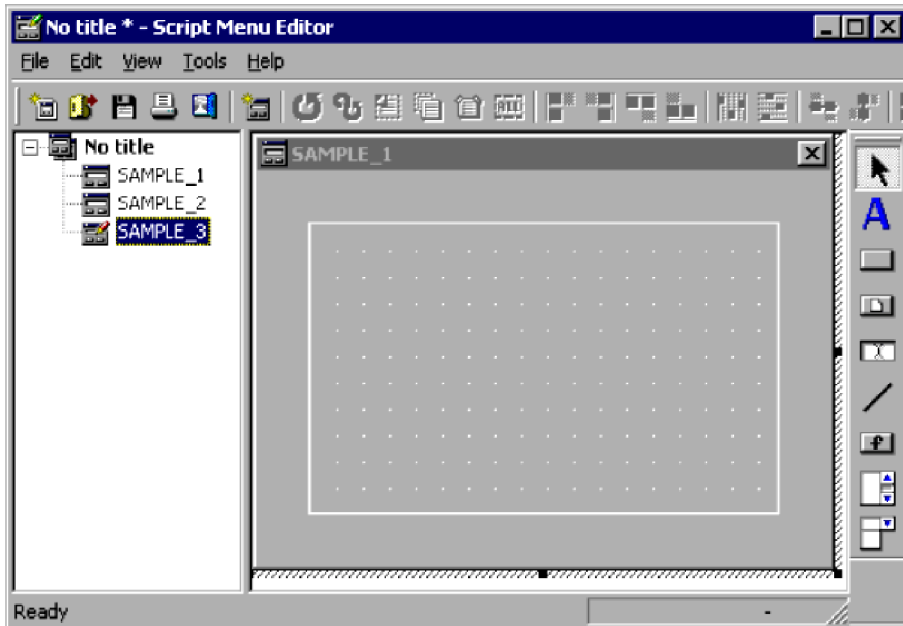
3. Set **Width**, **Height**, and **Margin**.

Set the values in pixels.



4. Click **OK**.

The grid appears on the menu form.



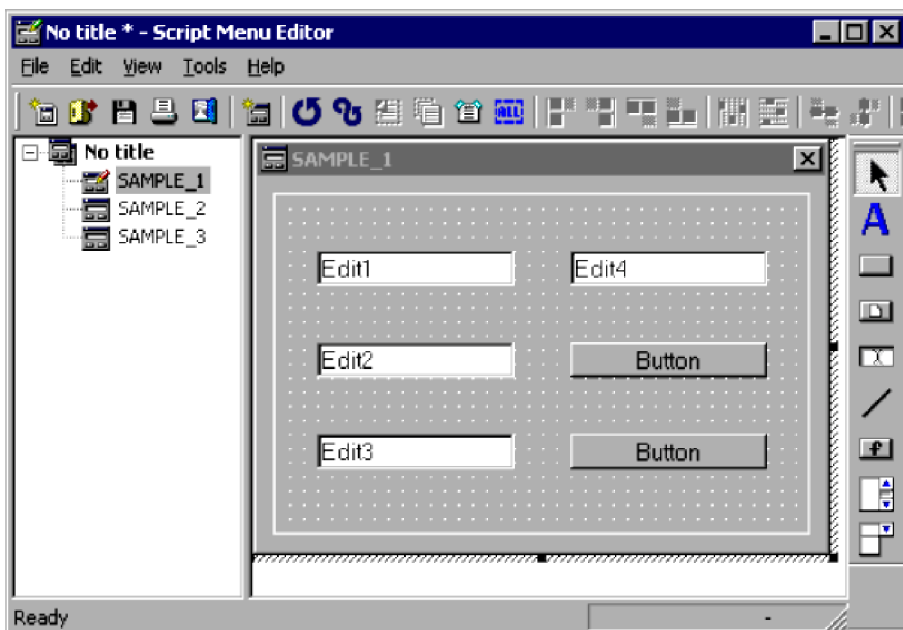
Notes

- When you drag and position a control on a menu form, the control snaps to the grid line. To place a control without using the grid, hold down the **Alt** key while you drag the control.
- By holding down the **Shift** key and dragging a control, you can move the control vertically and horizontally only.
- To remove the grid display, clear the **Use grid** check box.

3.6.14 Setting the tab order

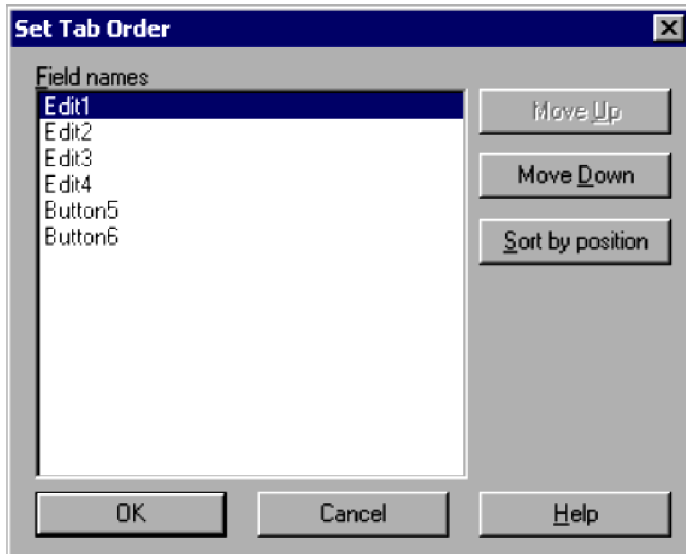
To set the order in which the tab key moves among controls:

1. Display a menu form.



2. Choose **Edit, Layout, Set Tab Order**.

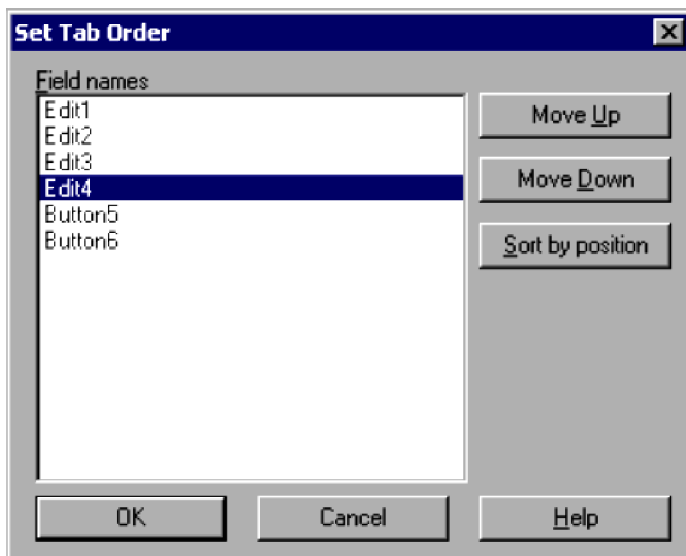
The Set Tab Order dialog box appears.



For details about using this dialog box, see [4.4.41 Set Tab Order dialog box](#).

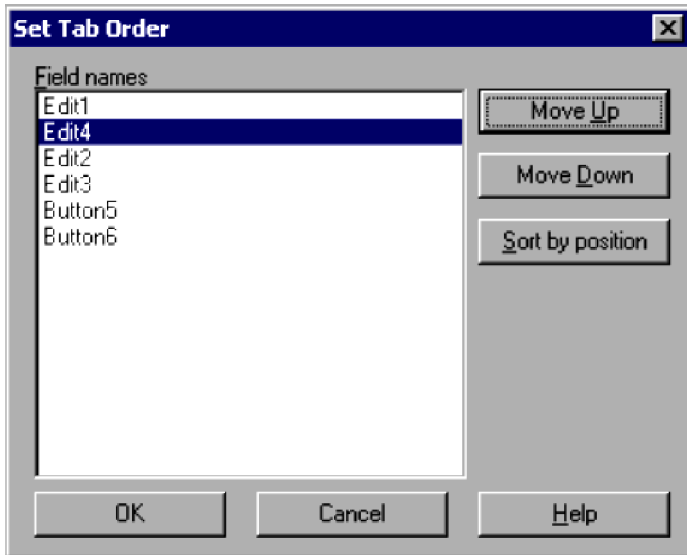
3. Select a field name.

Select a field name to move. In this example, **Edit4** is selected.



4. Move the field name.

Click the **Move Up** button to move the field up one line, or the **Move Down** button to move the field down one line. To sort the list in order of the field coordinates, click **Sort by position**.

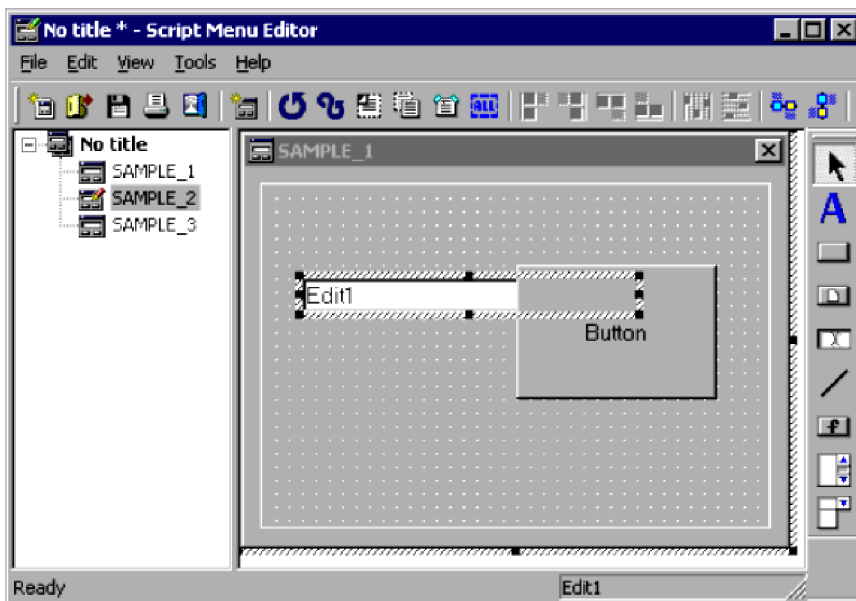


5. Click **OK**.

3.6.15 Moving a control to the foreground

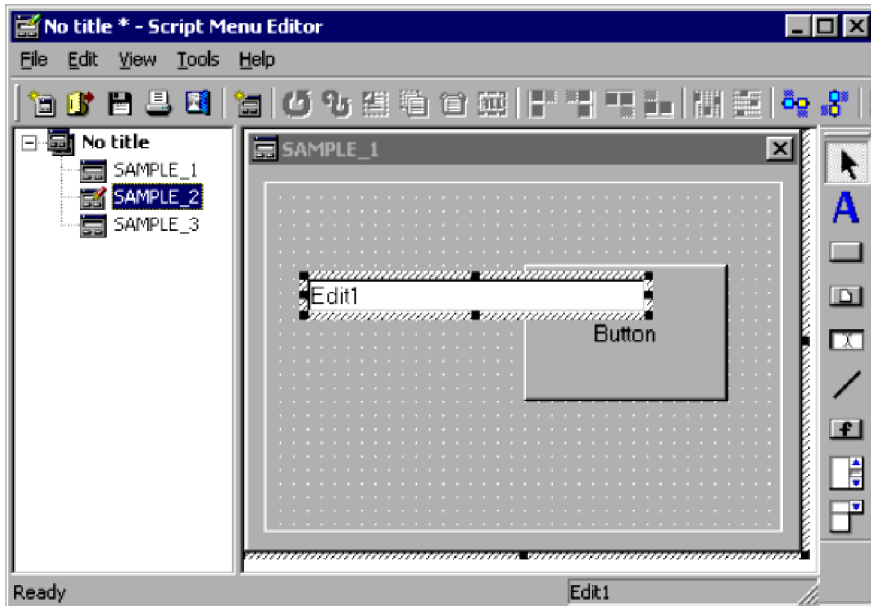
To move a selected control to the foreground:

1. Select a control on the menu form.



2. Choose **Edit, Layout, Move Forward**.

The selected control moves to the foreground.



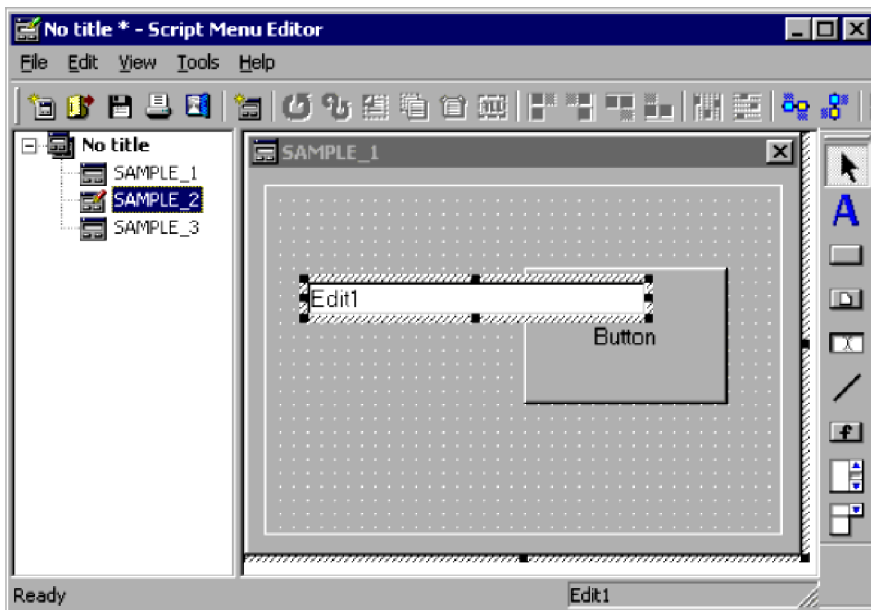
Note

The **Move Forward** command is disabled when no control is selected.

3.6.16 Moving a control to the background

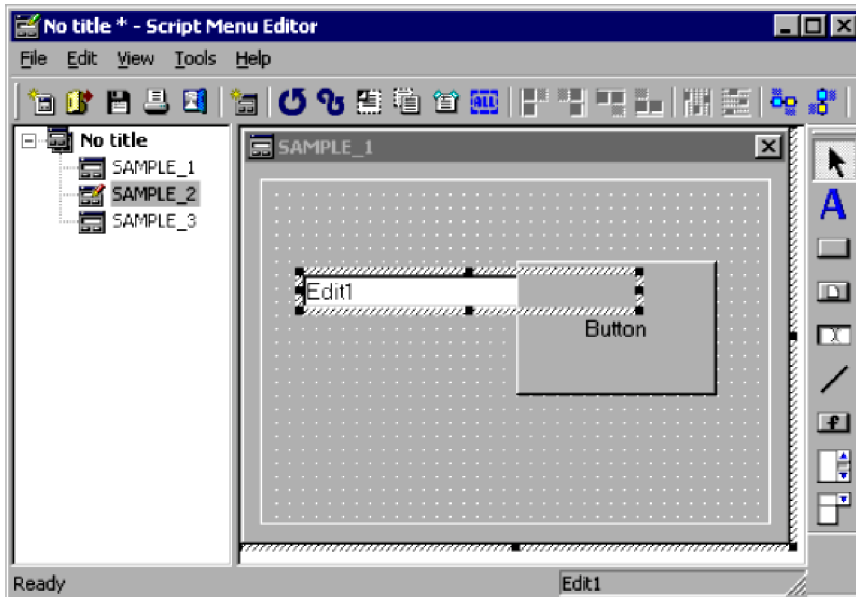
To move a selected control to the background:

1. Select a control on the menu form.



2. Choose **Edit, Layout, Move Backward**.

The selected control moves to the background.



Note

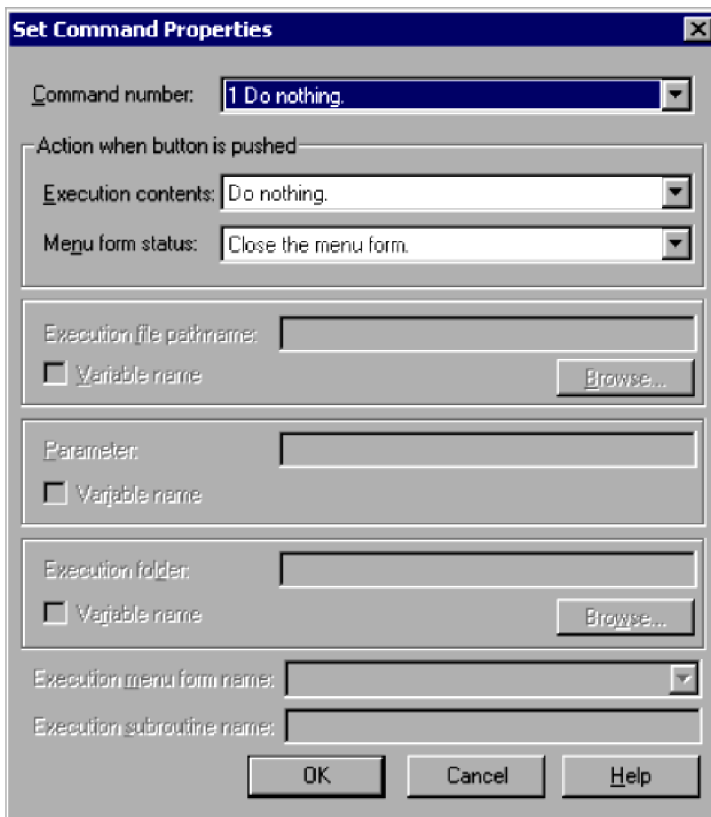
The **Move Backward** command is disabled when no control is selected.

3.6.17 Defining the properties of a command

To define command properties:

1. Choose **Tools, Set Command Properties**.

The Set Command Properties dialog box appears.



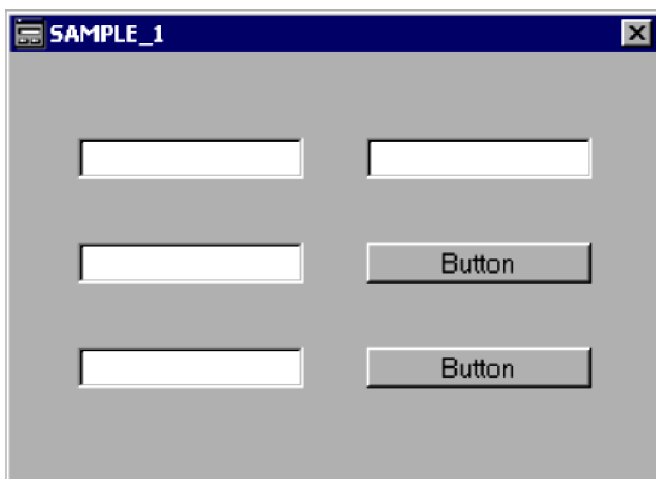
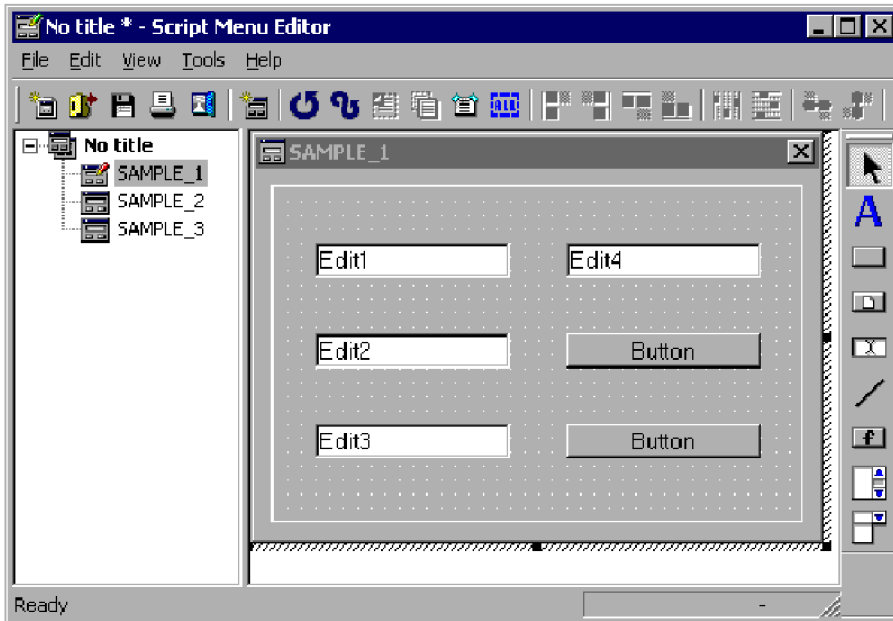
For details about using this dialog box, see *4.4.37 Set Command Properties dialog box*.

2. Entered the required information.
3. Click **OK**.
The command properties are set.

3.6.18 Displaying a menu form in test view

To display a menu form in test view:

1. Choose **Tools, Test View**.



The created menu form appears.

3.6.19 Printing a menu form

To print a menu form:

1. Choose **File, Print Menu Form**.

The Print Menu Form dialog box appears.

2. Enter the required information.

3. Click the **Print** button.

The definitions in the menu form are printed.

Note

If you click **Cancel** at step 3, the dialog box closes.

3.7 Process Viewer operations

Process Viewer allows you to monitor and control script processes that are running on the local computer and on remote computers. To monitor or control script processes running on a remote computer, you must log on as a user registered on the connection target computer.

Note

If you attempt to start Process Viewer while the JP1/Script service is not running, a dialog box containing the following warning appears and then Process Viewer starts:

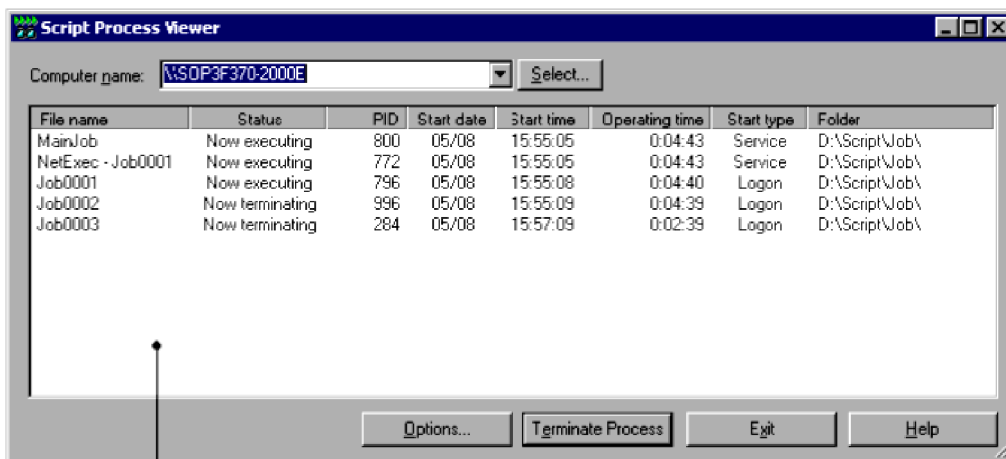
Processes that are started by a service or being executed by another user are not displayed because the JP1/Script service is not running.

3.7.1 Starting Process Viewer

To display the Script Process Viewer window, double-click the Process Viewer icon or choose **Tools, Start Process Viewer** in the Manager window.

Figure 3-8 shows the Script Process Viewer window.

Figure 3–8: Script Process Viewer window



Client area

(1) Computer name

Specify the computer to be connected. Choose **Select**, and then select a computer name from the displayed dialog box. If an error occurs during connection, an error message appears. A blank client area indicates that connection failed and the system is unconnected. The names of computers that were connected previously remain in the list box as a log.

(2) Client area

The client area lists information about script files (processes) running on the connected computer.

File name

Lists the script file names of active script processes. For example, the executable file called by the `NetExec` command is displayed in the form `NetExec - file-name`.

Status

Shows the execution status of the script process as either of the following:

Now executing: The process is now running.

Now terminating: The process is now terminating.

PID

Process ID of the script process.

Start date

Start date of the script process, displayed in the form *mm/dd* (*mm*: month; *dd*: day).

Start time

Start time of the script process, displayed in the form *HH:MM:SS* (*HH*: hour; *MM*: minute; *SS*: second).

Operating time

Time taken for the script process to run since the start time, displayed in the form *HH:MM:SS* (*HH*: hour; *MM*: minute; *SS*: second).

Start type

Start type of the script process as either of the following:

Logon: Started in the Logon space.

Service: Started in the Service space.

Folder

Folder name of the script file (current folder).

The Script Process Viewer window contains the following buttons:

Options

Sets the interval for refreshing the process view in the client area. Set 1 to 60 seconds.

Terminate Process

Terminates a script file (process) running on the connected computer.

From the client area, select a process you want to terminate, and then click **Terminate Process**. A dialog box for terminating the JP1/Script process appears. Choose **Yes** to terminate the process you selected.

If you want to disable this button, set the value 1 in the following registry key:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1/Script\SPTHView\Option

Value name

TermButton (DWORD)

Value datatype

REG_DWORD

Value

0: Enables the **Terminate Process** button.

1: Disables the **Terminate Process** button.

When the setting takes effect

The setting takes effect the next time Process Viewer is started.

Exit

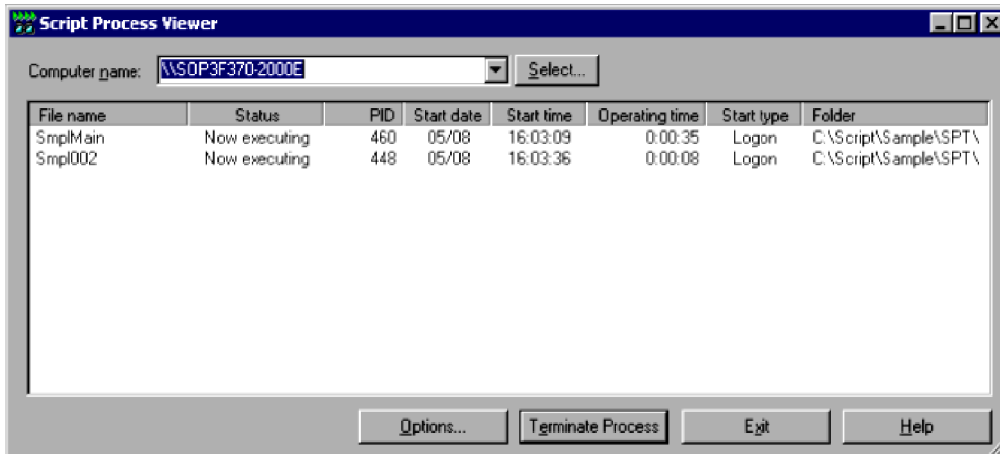
Closes the Script Process Viewer window.

Help

Displays the online help for the Script Process Viewer window.

3.7.2 Listing active script processes

In the Script Process Viewer window, you can list script processes running on the local computer or on a remote computer, and monitor and control the processes during execution.

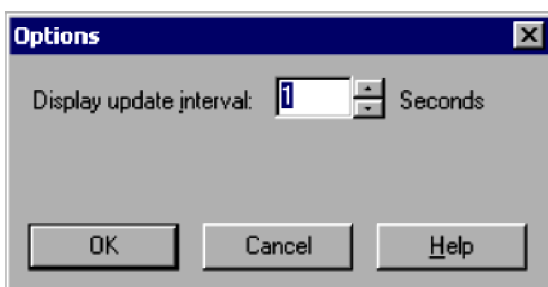


Operations

- If you specify a connected remote computer in **Computer name**, the script processes that are running on that computer are displayed.
- Click the **Options** button to display the Script Process Viewer Options dialog box. For details about using this dialog box, see [3.7.3 Specifying a refresh interval for listing script processes](#).
- To forcibly terminate one or more processes selected in the list, click the **Terminate Process** button.
- To close the Script Process Viewer window, click the **Exit** button.

3.7.3 Specifying a refresh interval for listing script processes

In the Script Process Viewer window, click the **Options** button to display the Options dialog box.



Display update interval

Specifies in seconds an interval for refreshing the script processes listed in the Script Process Viewer window. Set a value in the range 1 to 60. 1 is set by default.

Operations

- Click the **OK** button to apply the settings and close the Options dialog box.
- Click the **Cancel** button to close the Options dialog box without changing the original settings.

3.8 Execution Environment File Converter operations

3.8.1 Overview of Execution Environment File Converter

Execution Environment File Converter is a tool for exporting execution environment files to the UNIX version of JP1/Script. You use this program to create an execution environment syntax file from an execution environment file. An execution environment syntax file describes in text file format the contents of the execution environment file.

3.8.2 Converting an execution environment file

To export an execution environment file from Windows to UNIX:

1. Using Execution Environment File Converter, specify the execution environment file (.SPV) to be converted and execute the Converter.

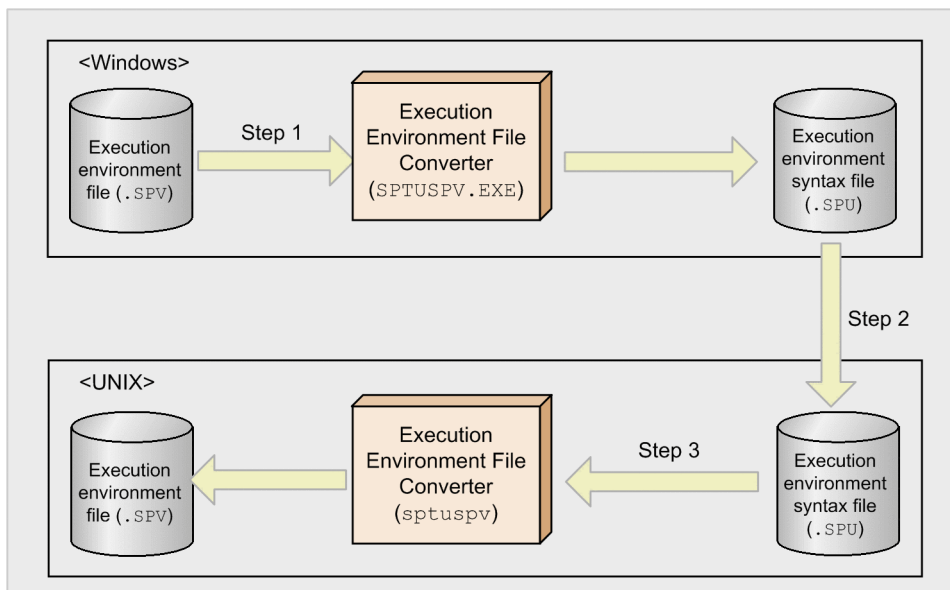
The Converter creates an execution environment syntax file (.SPU).

2. Move the created execution environment syntax file (.SPU) to UNIX by FTP transfer.

3. Using Execution Environment File Converter, specify the execution environment syntax file (.SPU) you moved in step 2, and then execute the Converter.

The execution environment file (.SPV) that can be used in UNIX is created.

Figure 3–9: Flow of exporting an execution environment file



Legend:

Execution environment file: A file in which the execution environment of a script is set

Execution environment syntax file: A file in which the execution environment of a script is set

3.8.3 Executing Execution Environment File Converter

To execute Execution Environment File Converter, start the command prompt, then on the command line enter a command in the format shown below.

(1) Converting an execution environment file (.SPV) to an execution environment syntax file (.SPU)

To execute this conversion, specify an execution environment file, which has the .SPV extension, as the source file for conversion.

Syntax (Δ : space)

```
sptuspv  $\Delta$  /i  $\Delta$  FromFileName (  $\Delta$  /o  $\Delta$  toDirName)
```

Arguments

FromFileName

Specify the execution environment file (.SPV) to be converted into an execution environment syntax file (.SPU). You can specify a wildcard in the file name.

toDirName

Specify the folder to which the converted environment execution syntax file (.SPU) is to be output. If you omit this specification, the file is output to the same folder containing the conversion source file. If a converted file already exists, all previous contents are lost.

(2) Converting an execution environment syntax file (.SPU) to an execution environment file (.SPV)

To execute this conversion, specify an execution environment syntax file, which has the .SPU extension, as the source file for conversion.

Syntax (Δ : space)

```
sptuspv  $\Delta$  /i  $\Delta$  fromFileName (  $\Delta$  /o  $\Delta$  toDirName) (  $\Delta$  operand)
```

Arguments

fromFileName

Specify the execution environment syntax file (.SPU) to be converted into an execution environment file (.SPV). You can specify a wildcard in the file name.

toDirName

Specifies the folder to which the converted execution environment file (.SPV) is to be output. If this specification is omitted, the converted file is output to the same folder containing the conversion source file. If a converted file already exists, all previous contents are lost.

operand

You can specify the following operand:

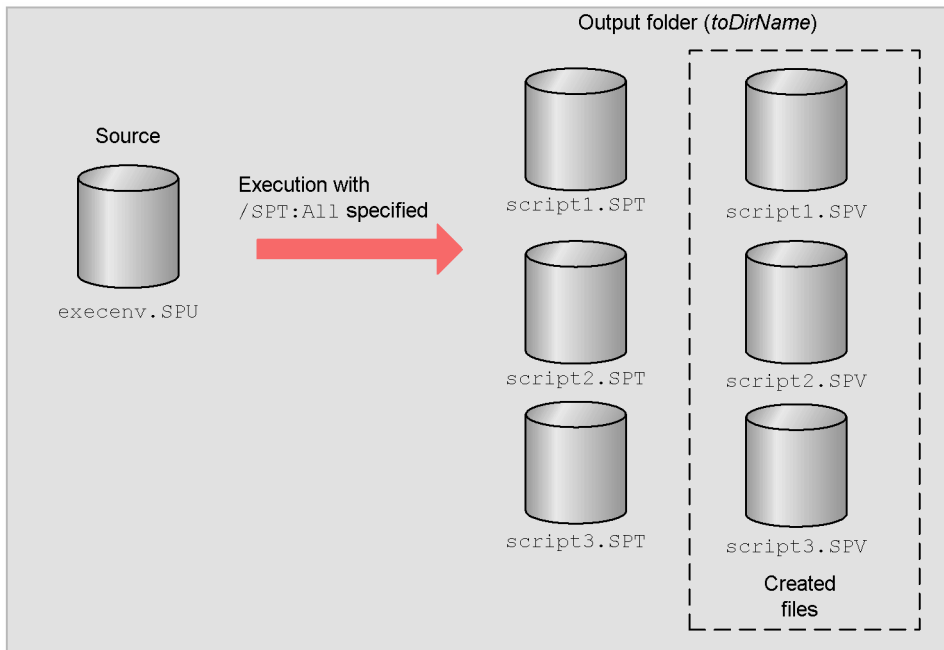
`/SPT:ALL`

Specify this operand if you want to apply the conversion from one execution environment syntax file (.SPU) to all script files (.SPT) found in the output destination folder. Specify the operand after *output-target-directory-name*, or specify the operand after *source-file-name* when *output-target-directory-name* is omitted.

If you specify `/SPT:ALL`, you cannot specify a wildcard in the conversion source file name (if you do, an error results).

Figure 3-10 shows the flow of creating an execution environment file when `/SPT:ALL` is specified.

Figure 3–10: Flow of creating an execution environment file when /SPT:ALL is specified



If you execute Script Execution Environment File Converter with `execenv.SPU` specified as the source file, `Dir` specified as the output folder, and the `/SPT:ALL` operand specified, the execution environment files `script1.SPV` to `script3.SPV` are created based on the contents of `execenv.SPU` for the files `script1.SPT` to `script3.SPT` at the output target. If you omit the `/SPT:ALL` operand, only `execenv.SPU` is created in the output folder.

Return value

If conversion terminates normally, 0 is returned; if conversion does not terminate normally, an error code is returned.

Displayed message

The command `sptuspv` displays the messages shown in Table 3-24 on the console. If you execute the Converter without a console, allocate a standard output when executing the Converter, and then check the message. Messages are displayed separately for each file conversion.

If an execution environment file or execution environment syntax file contains an error, the Converter stops conversion of that file. If a wildcard is specified in the file name for `fromFileName`, the Converter continues conversion of any file that has not yet been converted.

Table 3–24: Return values and messages of Execution Environment File Converter

No.	Return value	Meaning of return value	Message
1	0	Conversion terminated normally (if a wildcard was specified, conversion of all target files terminated normally).	The file has been converted. (name-of-converted-file)
2	1	The syntax of the execution environment syntax file contains an error.	There is a syntax error in the execution environment syntax file. (name-of-file-affected-by-error)
3	2	An out-of-range number was specified.	A value outside the range has been specified in the execution environment syntax file. (name-of-file-affected-by-error)
4	3	A file's format is incorrect.	The input file is damaged or has a different file format. (name-of-file-affected-by-error)
5	--	The file to be converted already exists in the output destination folder.	The file after conversion already exists on the output destination directory. The file contents have been lost completely. (name-of-overwritten-file)

No.	Return value	Meaning of return value	Message
6	4	The input file does not exist.	System error message. (name-of-file-affected-by-error)
7	5	An input file access error occurred.	File management information cannot be read. (name-of-file-affected-by-error)
8	20	The output destination directory cannot be found.	The output destination directory was not found. (name-of-directory-affected-by-error)
9	21	An output file access error occurred.	System error message. (name-of-file-affected-by-error)
10	22	A file that is neither the execution environment file (.SPV) nor the execution environment syntax file (.SPU) was specified.	The file cannot be specified. (name-of-file-affected-by-error)
11	23	No parameters were specified, or a specified value is incorrect.	An invalid parameter was specified.
12	24	An operand is invalid.	An invalid combination of parameters was specified.
13	25	There is insufficient memory.	A memory shortage occurred.
14	26	The target script file does not exist.	The targeted script file did not exist in the output destination directory.
15	27	The specified directory cannot be found.	The directory specified in the execution environment syntax file was not found. (name-of-directory-affected-by-error)
16	--	An error occurred when the output file was being overwritten.	The file was deleted because a write error occurred for the existing file. (name-of-file-affected-by-error)
17	99	Partial conversion failed.#	--

#

- This value is returned if a wildcard is specified for the input file and No. 2, 3, 4, 6, or 7 occurs.
- This value is returned if the current file cannot be converted but another file was converted.
- If an error other than No. 2, 3, 4, 6, or 7 occurs, processing is interrupted at that point.

3.8.4 Overview of execution environment syntax files (.SPU)

An execution environment syntax file (.SPU) is a text file that shows the contents of an execution environment file that contains the environment settings for running scripts.

Table 3-25 provides an overview of the execution environment syntax files.

Table 3–25: Overview of execution environment syntax files

File type	File name	Extension	File contents
File handled by the Converter	Execution environment syntax file	.SPU	This file shows the contents of an execution environment file in text format. It enables you to use a text editor to set the execution environment.

3.8.5 Details of execution environment syntax files (.SPU)

Figure 3-11 shows the format of an execution environment syntax file.

Figure 3–11: Format of an execution environment syntax file

```

Start_CommandLine=/ABC/123    ... 1.
Start_Work_Dir=C:\Temp       ... 2.
Start_MultiRun=0             ... 3.
Start_Type=0                 ... 4.
Start_Date_Appo=0           ... 5.
Start_Week_Appo=0           ... 6.
Start_Week=1                 ... 7.
Start_Day_Appo=0            ... 8.
Start_Day=1                  ... 9.
Start_TimeAppo=0            ... 10.
Start_Time_Hour=9           ... 11.
Start_Time_Minuts=00        ... 12.
End_StopTimes=0            ... 13.
Trace_Dir=C:\temp           ... 14.
Trace_MaxLines=1024         ... 15.
Trace_MaxColumns=150        ... 16.
Trace_user_MaxLines=1024    ... 17.
Trace_user_MaxColumns=150   ... 18.

```

Description

1. Command line.

If the string indicating whether or not a trace file or a log file is to be output is specified in Windows format, the string is converted to UNIX format.

2. Work folder. When an execution environment file is converted to an execution environment syntax file, always set a blank for this item. When an execution environment syntax file is converted to an execution environment file, this value becomes effective. If a nonexistent folder name is specified, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

3. Multiple startups enable. Specify 1 to allow multiple startups, or 0 to not allow multiple startups. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

4. Startup type. Specify 0 for logon, or 1 for service. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

This item is valid only for the Windows version of Script Converter.#

5. Specify whether or not you are specifying the date for automatic startup. Specify 1 if you are specifying the date, or 0 if you are not. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

This item is valid only for the Windows version of Script Converter.#

6. Day of the week specification. Specify 1 if you are specifying the day of the week, or 0 if you are not. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

This item is valid only for the Windows version of Script Converter.#

7. Day of the week. Specify a value from 1 to 7. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

This item is valid only for the Windows version of Script Converter.#

Table 3–26: Day of the week corresponding to specified value

Value	Day of the week
1	Monday
2	Tuesday

Value	Day of the week
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

8. Date specification. Specify 1 if you are specifying the date, or 0 if you are not. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
This item is valid only for the Windows version of Script Converter.#
9. Date. Specify a value from 1 to 31. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file and the execution environment file is not created.
This item is valid only for the Windows version of Script Converter.#
10. Time specification. Specify 1 if you are specifying the time, or 0 if you are not. If you specify any other value, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
This item is valid only for the Windows version of Script Converter.#
11. Time (hour). Specify a value from 0 to 23. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
This item is valid only for the Windows version of Script Converter.#
12. Time (minute). Specify a value from 0 to 59. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
This item is valid only for the Windows version of Script Converter.#
13. End time. Specify the number of minutes that can elapse before the script file being executed is forcibly terminated. Specify a value from 0 to 1,440. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created. If 0 is specified, the system assumes that the end time is not specified.
This item is valid only for the Windows version of Script Converter.#
14. Output destination folder for trace. If you specify a nonexistent folder, an analysis error occurs. When an execution environment file is converted to an execution environment syntax file, always set a blank for this item. When an execution environment syntax file is converted to an execution environment file, this value becomes effective. If a nonexistent folder name is specified, a syntax error occurs during conversion to the execution environment file and the execution environment file is not created.
15. Maximum number of trace lines. Specify a value from 100 to 9,999. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
16. Maximum number of trace columns. Specify a value from 128 to 1,024. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.
17. Maximum number of user trace rows. Specify a value from 100 to 9,999. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

18. Maximum number of user trace columns. Specify a value from 128 to 1,024. If you specify a value outside this range, a syntax error occurs during conversion to the execution environment file, and the execution environment file is not created.

#: If this item is specified in the UNIX version of an execution environment syntax file, the default value is used during conversion from the execution environment file (.SPV).

Other

The description rules for execution environment syntax files are as follows:

- Any items can be omitted, and the default value is assumed for the items that are not specified.
- If the same item is specified more than once, the value specified last is effective.
- The items can be specified in any order.
- Any line that begins with # is treated as a comment line.

4

JP1/Script Dialog Boxes

This chapter provides detailed descriptions of the dialog boxes that open from the Script Manager window, Script Editor window, Script Trace Viewer window, and Script Menu Editor window.

4.1 Script Manager dialog boxes

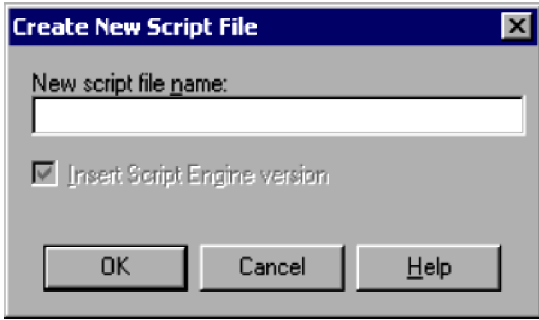
The dialog boxes displayed when you use the Script Manager window are listed below. Detailed descriptions of each dialog box and its components are given after this list.

- Create New Script File dialog box
- Copy Files dialog box
- Add Files dialog box
- Confirm File Overwrite dialog box
- Confirm File Deletion dialog box
- Rename dialog box
- Execution dialog box
- Set Execution Environment (Start Information) dialog box
- Set Execution Environment (Terminate Information) dialog box
- Set Execution Environment (Trace Information) dialog box
- Set Execution Environment (Command Line) dialog box
- Set Command Line dialog box
- Set Execution Environment (Working Folder) dialog box
- Set Execution Environment (Start Date) dialog box
- Set Execution Environment (Terminate Time) dialog box
- Set Execution Environment (Trace Output Folder) dialog box
- Set Execution Environment (User Trace Information) dialog box
- Change Folder dialog box
- File Properties dialog box
- Set Automatic Startup dialog box
- Link Editor dialog box
- Options (Server Information) dialog box
- Update Information dialog box
- Options (Compatibility) dialog box
- Options (Multi-activation) dialog box
- Options (JP1/IM) dialog box
- Options (Trace) dialog box
- Options (Cluster Environment) dialog box
- Select Folder dialog box

4.1.1 Create New Script File dialog box

The Create New Script File dialog box appears when you choose **File, Create** in the Script Manager window.

Enter the name for the script file you want to create.



(1) Components

New script file name

Enter the name of the script file to be created.

Insert Script Engine version

Choose whether to insert the version of Script Engine to be used for file execution at the head of the script file. This setting takes effect only when an editor other than Editor has been associated in the Link Editor dialog box.

(2) Operations

- Enter a script file name and then click **OK** to launch the editor specified in the Link Editor dialog box (**Tools, Link Editor** command).
- Click **Cancel** to cancel script creation.

(3) Processing

- File names must comply with the naming conventions of the Windows file system, except that you cannot use spaces or the following characters:

```
= : , . { }
```

- File naming depends on the network operating system. Comply with the naming conventions for the file system of the operating system you are using.
- If no editor is specified in the Link Editor dialog box, a message to that effect is displayed and processing is canceled.
- After creating a script file with the dedicated Editor, you must choose **View**, and then **Refresh** to apply the created file to the client area of the Script Manager window.
- When you select the **Insert Script Engine version** check box, the following line is inserted at the head of the script file:

```
#FileVersion=VVRR (VV: JP1/Script version; RR: JP1/Script revision)
```

The version and revision of the installed JP1/Script program is set in *VVRR*.

If you do not select the **Insert Script Engine version** check box, the script file will be executed using the version set in the Options (Compatibility) dialog box of Script Manager.

Important note

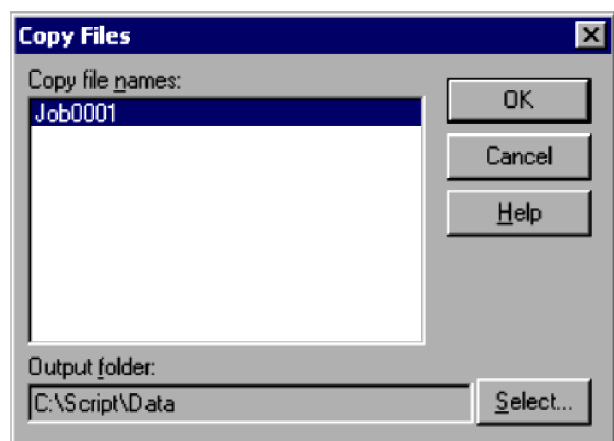
- You cannot use the standard Windows WordPad as an editor for creating a script. Use another editor that supports saving in text format.

- The Script Engine version does not need to be the same as the version of the installed JP1/Script program. You can apply the conventions of the version of Script Engine to execute a script file. For details, see *6.1.11 Script coding conventions*.

4.1.2 Copy Files dialog box

The Copy Files dialog box appears when you select an icon in the Script Manager window and choose **File, Copy**.

Select script files to copy from the current folder to another folder.



(1) Components

Copy file names

Lists one or more files names that you selected in the Manager window.

Output folder

Enter the name of a folder to copy the file(s) to. If you click the **Select** button, the Select Folder dialog box appears.

Select

Displays the Select Folder dialog box for selecting a destination folder.

(2) Operations

- Enter a folder name, then select one or more file names from the list to copy to this folder. You can also click **Select** to choose a destination folder if you wish.
- Click **OK** to copy the file(s). Click **Cancel** to cancel file copying.

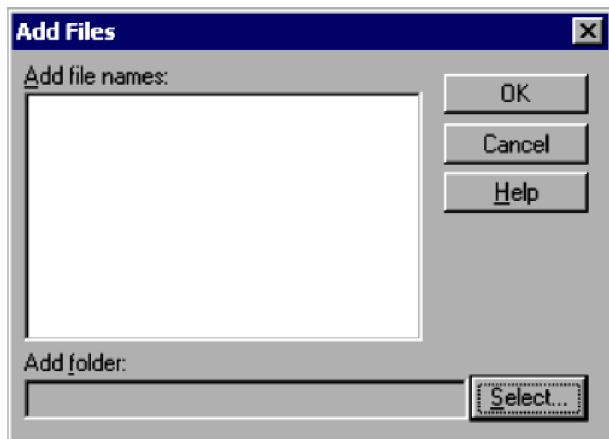
(3) Processing

- A dialog box shows copying progress.
- If a file of the same name already resides in the destination folder, the Confirm File Overwrite dialog box appears.

4.1.3 Add Files dialog box

The Add Files dialog box appears when you select an icon in the Script Manager window and choose **File, Add**.

Use this dialog box to add a script file from another folder to the current folder.



(1) Components

Add file names

Lists the script files in the source folder.
Select one or more file names to add.

Add folder

Enter the name of the folder to copy file(s) from.

Select

Displays the Select Folder dialog box for selecting a source folder.

(2) Operations

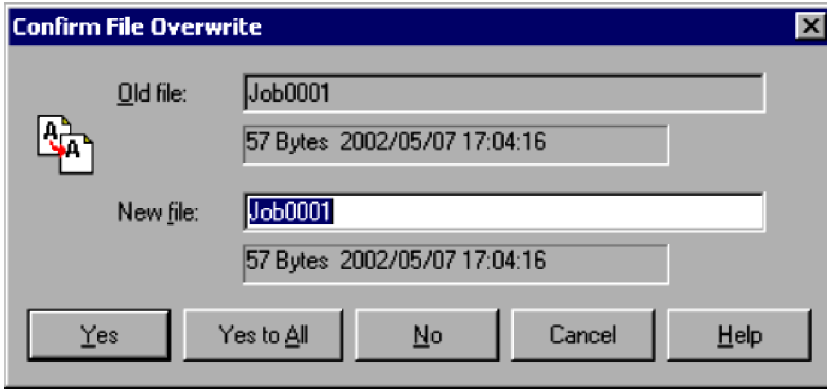
- Enter a folder name, then select one or more files to add from this folder. You can also click **Select** to choose a source folder if you wish.
- Click **OK** to add the file(s). Click **Cancel** to cancel file addition.

(3) Processing

- A dialog box shows the progress of file addition.
- If a file of the same name already resides in the destination (current) folder, the Confirm File Overwrite dialog box appears.

4.1.4 Confirm File Overwrite dialog box

The Confirm File Overwrite dialog box appears if copying or adding a file means that the contents of a file will be replaced.



(1) Components

Old file

Shows the name of a file you are copying or adding. You cannot change this name.

The file size, and the date and time the file was last modified, appear under the file name.

New file

Shows the name of the file that will be overwritten. The size and the update date and time of the file that will be overwritten appear under the file name.

If you do not want to overwrite the file, specify a new file name, and then click **Yes**.

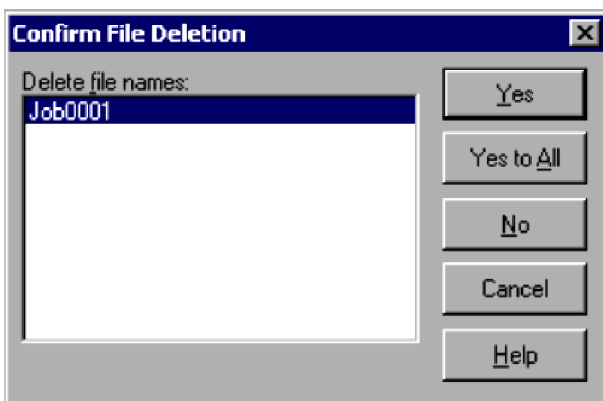
(2) Operations

- To replace the file contents, click **Yes** or **Yes to All**. If you click **Yes to All**, this dialog box will be skipped for the subsequent files.
- Click **No** to continue processing without replacing the file contents.
- Click **Cancel** to cancel processing.

4.1.5 Confirm File Deletion dialog box

The Confirm File Deletion dialog box appears when you select an icon in the Script Manager window and choose **File, Delete**.

Use this dialog box to delete the created script file.



(1) Components

Delete file names

Lists one or more files names that you selected in the Script Manager window.

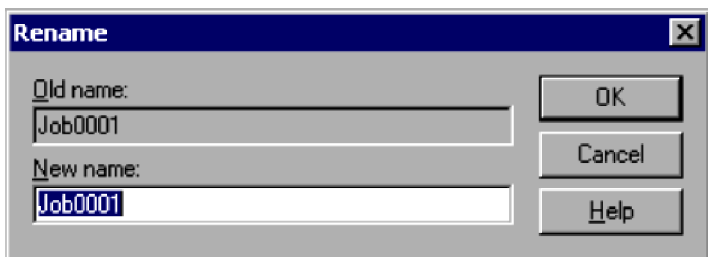
(2) Operations

- From the list of files to delete, select one or more file names. Then click **Yes** to delete the first file and erase its icon from the client area in the Script Manager window. If you selected multiple files, this processing is repeated for each file in turn.
- Click **Yes to All** to delete all the file names in the list.
- Click **No** to keep the file. If you selected multiple files, this processing is repeated for each file in turn.
- Click **Cancel** to cancel file deletion.

4.1.6 Rename dialog box

The Rename dialog box appears when you choose **File, Rename**.

Use this dialog box to rename a file selected in the Script Manager window.



(1) Components

Old name

Shows the name of the file (icon) that you selected in the Script Manager window.

New name

Shows the new name.

(2) Operations

- To rename the file, enter the new name and then click **OK**.
- Click **Cancel** to close the dialog box without renaming the file.

(3) Processing

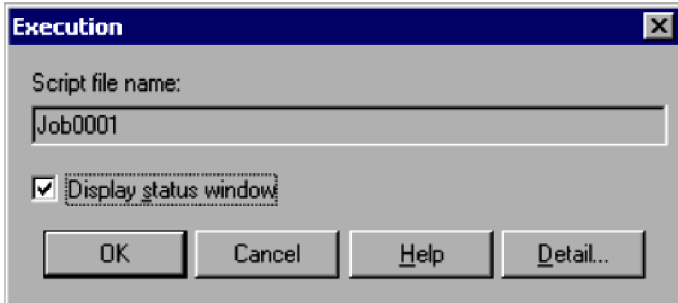
- The file name must comply with the naming conventions of the Windows file system, except that you cannot use spaces or the following characters:

= : , . { }

- File naming depends on the network operating system. Comply with the naming conventions for the file system of the operating system you are using.

4.1.7 Execution dialog box

The Execution dialog box appears when you select an icon in the Script Manager window and choose **File, Execute**.



(1) Components

Script file name

Shows the name of the file (icon) that you selected in the Script Manager window.

Display status window

Select whether to display windows showing the analysis trace, execution trace, and execution status.

(2) Operations

- Click **OK** to execute the script.
- Click **Cancel** to cancel script execution.
- Click **Detail** to set the execution environment. The settings are entered in an execution environment file. For information on setting the execution environment, see [4.1.8 Set Execution Environment \(Start Information\) dialog box](#), [4.1.9 Set Execution Environment \(Terminate Information\) dialog box](#), and [4.1.10 Set Execution Environment \(Trace Information\) dialog box](#).

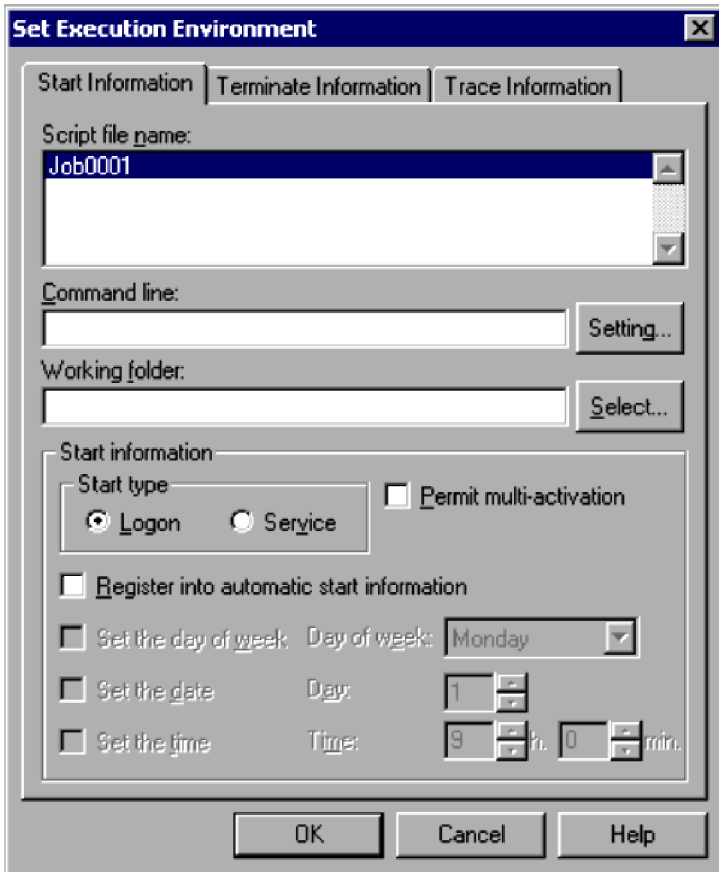
(3) Processing

- Windows showing the script execution status appear during execution.

4.1.8 Set Execution Environment (Start Information) dialog box

The Set Execution Environment dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, All Items**. It also appears when you click the **Detail** button in the Execution dialog box.

The Set Execution Environment dialog box has three tabs: **Start Information**, **Terminate Information**, and **Trace Information**. Click the **Start Information** tab to display the Set Execution Environment (Start Information) page.



Note

If you attempt to set up the execution environment (start information) while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

In the Set Execution Environment (Start Information) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Command line

Enter the run-time command line. You can specify a parameter, and whether to output a trace file and event log. If you click the **Set** button, the Set Command Line dialog box appears. For details, see [6.2 Rules for writing command lines](#).

Working folder

Enter the current drive or folder for script execution. If you click the **Select** button, the Select Folder dialog box appears. If you do not specify a folder, the folder containing the script file is selected.

Start information

Specify the target space where the script is to be started automatically. If no day of week, day, or time is set, the script is started at the following time:

Logon: When the OS is logged on

Service: When the power is turned on

If the OS is not logged on, the script is not started even when the target is logged on and a day of week, day, or time has been set.

Permit multi-activation

Select this check box to allow multiple copies of the same script to be executed concurrently.

There is no limit to the number of multiple executions; however, as the number of concurrent executions increases, the amount of required system resources also increases. You should pay attention to the number of scripts that are executed concurrently.

When multiple execution is enabled, traces are output to the folder specified in **Trace file output folder** in the Options (Multi-activation) dialog box, not the folder specified in the Set Execution Environment (Trace Information) dialog box. For further details, see [4.1.25 Options \(Multi-activation\) dialog box](#).

You cannot specify the trace file output folder for each script file. Therefore, if you specify **Permit multi-activation** for a script file, for each concurrent execution of that script, analysis trace files and execution trace files that have the same name are output to the same folder.

Register into automatic start information

Starts the script automatically at the Script Launcher startup.

Set the day of week[#]

Starts the script on a particular day of the week.

This check box is available only if you select **Register into automatic start information**.

Day of week[#]

Specify a day of the week for weekly execution.

This box is available only if you select **Set the day of week**.

Set the date[#]

Starts the script on a particular day of the month.

This check box is available only if you select **Register into automatic start information**.

Day[#]

Specify a day of the month for monthly execution. The script will not start in a month that does not have the specified day.

This box is available only if you select **Set the date**.

Set the time[#]

Starts the script at a particular time.

This check box is available only if you select **Register into automatic start information**.

Time[#]

Specify the hour and minute for script execution. Enter a value between 00:00 and 23:59. If you specify a time earlier than the Script Launcher start time, the script will run the following day.

This box is available only if you select **Set the time**.

#

This information takes effect only after the JP1/Script service or the Script Launcher is started.

(2) Operations

- Set information about the execution environment as required.
- Click **OK** to create the settings as an execution environment file.
- Click **Cancel** to close the dialog box without creating an environment execution file.

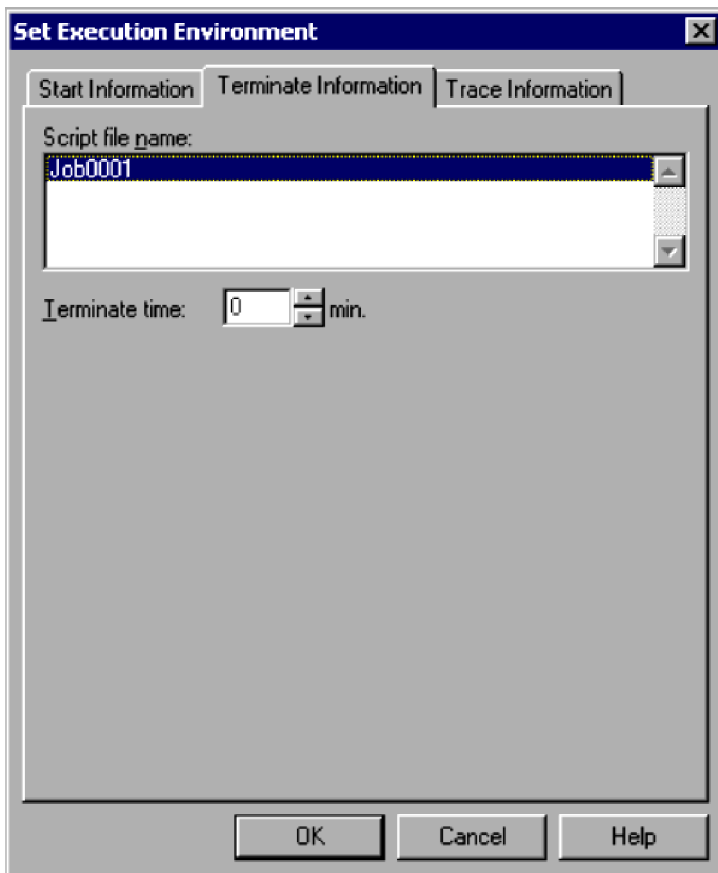
(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.
- If you omit the day of the month and time settings, the script is executed when the JP1/Script service or the Script Launcher starts.

4.1.9 Set Execution Environment (Terminate Information) dialog box

The Set Execution Environment dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, All Items**. It also appears when you click the **Detail** button in the Execution dialog box.

The Set Execution Environment dialog box has three tabs: **Start Information**, **Terminate Information**, and **Trace Information**. Click the **Terminate Information** tab to display the Set Execution Environment (Terminate Information) page.



Note

If you attempt to set up the execution environment (terminate information) while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

In the Set Execution Environment (Terminate Information) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Terminate time

Specify the run time in minutes before the script file is forcibly terminated.

Enter a number in the range 0 to 1,440. If you specify zero, the script file will not be forcibly terminated.

(2) Operations

- Set information about the execution environment as required.
- Click **OK** to create the settings as an execution environment file.
- Click **Cancel** to close the dialog box without creating an environment execution file.

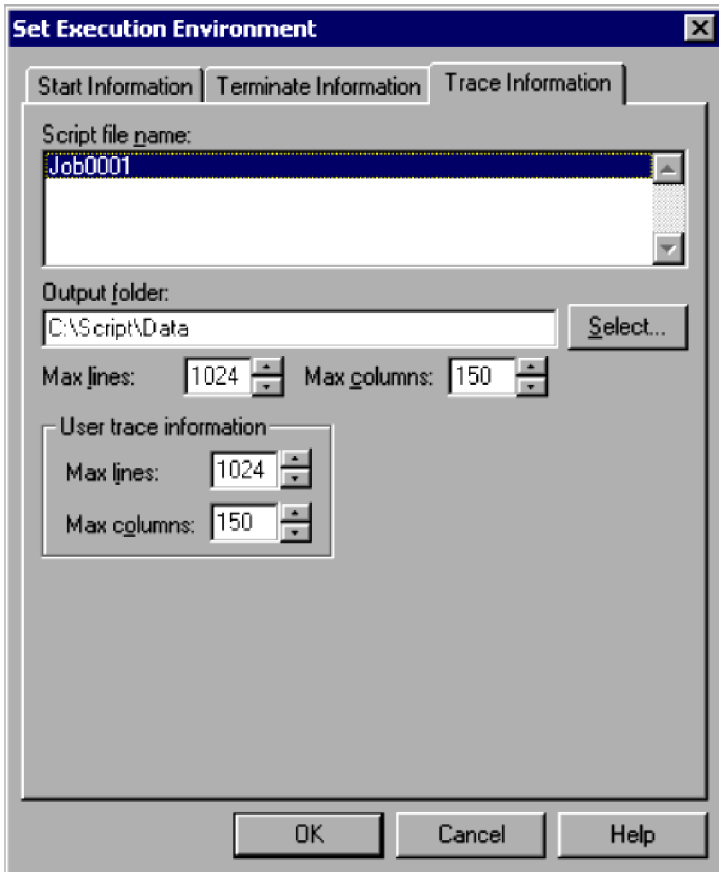
(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.
- By entering a termination time, you can enable a process to be automatically terminated if, for some reason, it fails to complete within a particular time.

4.1.10 Set Execution Environment (Trace Information) dialog box

The Set Execution Environment dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, All Items**. It also appears when you click the **Detail** button in the Execution dialog box.

The Set Execution Environment dialog box has three tabs: **Start Information**, **Terminate Information**, and **Trace Information**. Click the **Trace Information** tab to display the Set Execution Environment (Trace Information) page.



Note

If you attempt to set up the execution environment (trace information) while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

In the Set Execution Environment (Trace Information) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Output folder

Specify a folder for the analysis trace file and execution trace file.

If you specify a folder that does not exist, the folder is created when the script file starts.

If you click the **Select** button, the Select Folder dialog box appears.

The entry box is available only if you select the **Permit multi-activation** check box in the Set Execution Environment (Start Information) dialog box.

Max lines

Specify the maximum number of lines to output from the analysis trace file and execution trace file. Set a number in the range 100 to 9,999.

Max columns

Specify the maximum number of columns to output from the analysis trace file and execution trace file. Set a number in the range 128 to 1,024.

User trace information

Max lines

Specify the maximum number of lines to output from the user trace file. Set a number in the range 100 to 9,999.

Max columns

Specify the maximum number of columns to output from the user trace file. Set a number in the range 128 to 1,024.

(2) Operations

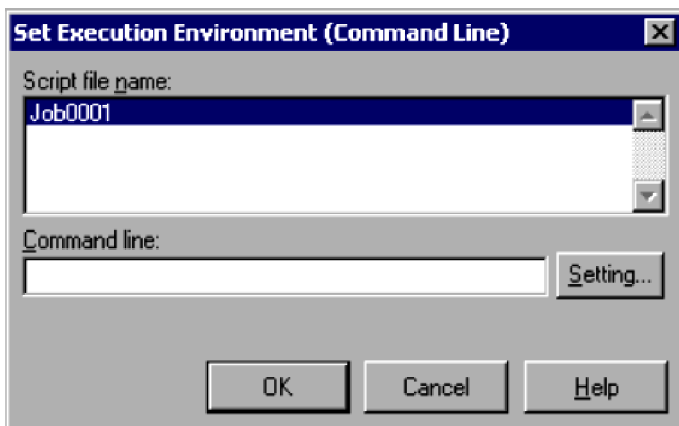
- Set information about the execution environment as required.
- Click **OK** to create the settings as an execution environment file.
- Click **Cancel** to close the dialog box without creating an environment execution file.

(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.

4.1.11 Set Execution Environment (Command Line) dialog box

The Set Execution Environment (Command Line) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, Command Line**.



(1) Components

In the Set Execution Environment (Command Line) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Command line

Specify the run-time command line. You can specify a parameter, and whether to output a trace file and event log. If you click the **Setting** button, the Set Command Line dialog box appears. For details on writing command lines, see *6.2 Rules for writing command lines*.

(2) Operations

- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the entered setting.
- Click **Cancel** to close the dialog box without updating the environment execution file.

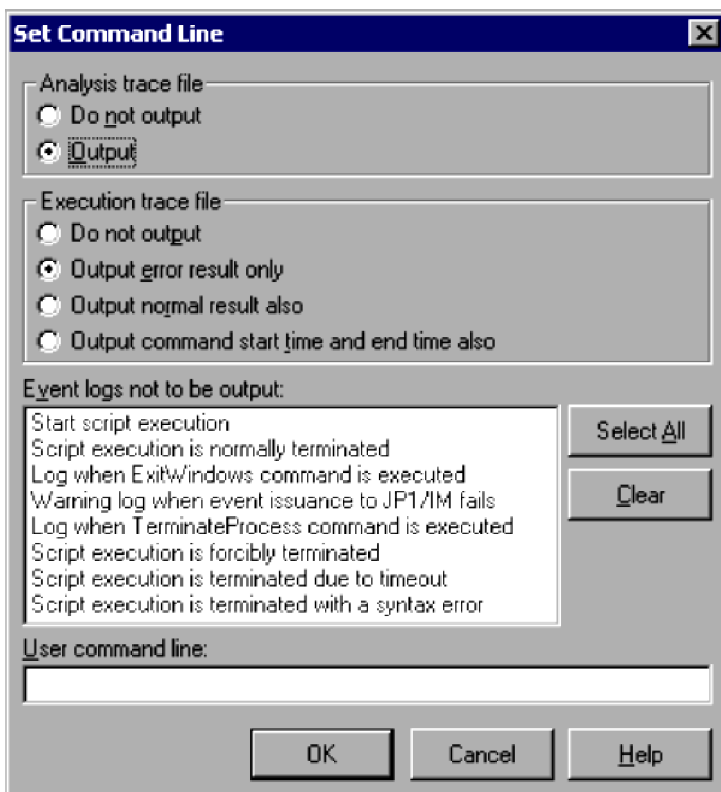
(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.

4.1.12 Set Command Line dialog box

The Set Command Line dialog box appears when you click the **Setting** button in the Set Execution Environment (Start Information) dialog box or Set Execution Environment (Command Line) dialog box.

Enter the run-time command line.



(1) Components

Analysis trace file

Choose whether to output an analysis trace file. **Output** is set by default.

The setting you choose here is set in the form[#] `/SPALV (n)` in the command line in the Set Execution Environment (Start Information) dialog box and Set Execution Environment (Command Line) dialog box.

Execution trace file

Choose whether to output an execution trace file and the output level. **Output error result only** is set by default.

The setting you choose here is set in the form[#] `/SPXLV (n)` in the command line in the Set Execution Environment (Start Information) dialog box and Set Execution Environment (Command Line) dialog box.

If you select **Output normal result also** or **Output command start time and end time also**, the execution performance of script files will be degraded because the amount of output information will increase.

Event logs not to be output

Lists event logs for which you can disable run-time output.

Click **Select All** to select all the event logs. Click **Clear** to clear the selection.

By default, the event logs are not all selected.

The settings you choose here are set in the form `/NOEVLOG` or `/NOEVLOG (n, n, . . .)` in the command line in the Set Execution Environment (Start Information) dialog box and Set Execution Environment (Command Line) dialog box.

User command line[#]

Enter a user-specified command line.

[#]

For details about the command format, see [6.2.2 Command line parameters](#).

(2) Operations

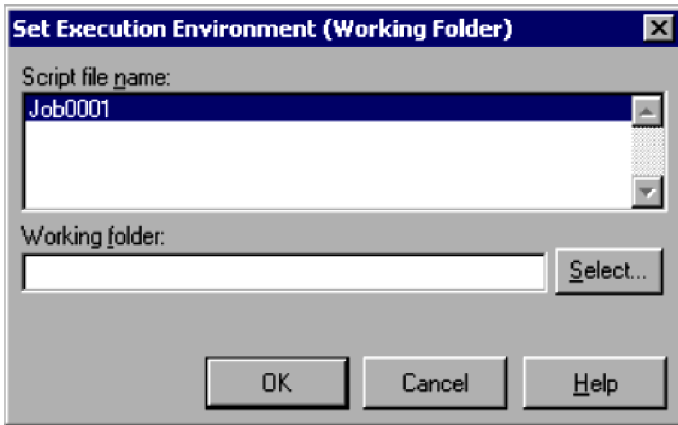
- Set command line information as required.
- Click **OK** to enter the settings in the command line displayed in the Set Execution Environment (Start Information) dialog box.
- Click **Cancel** to close the dialog box without changing the command line contents.

(3) Processing

- The **Event logs not to be output** list contains run-time event logs other than error logs.

4.1.13 Set Execution Environment (Working Folder) dialog box

The Set Execution Environment (Working Folder) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, Working folder**.



(1) Components

In the Set Execution Environment (Working Folder) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Working folder

Specify the current drive or folder for script execution.

If you do not specify a folder, the folder containing the script file is selected.

Note

Use the `SetPath` command to set the current folder for execution (work folder).

For details about the `SetPath` command, see [8.5.28 SetPath \(set the path to the executable folder\)](#).

(2) Operations

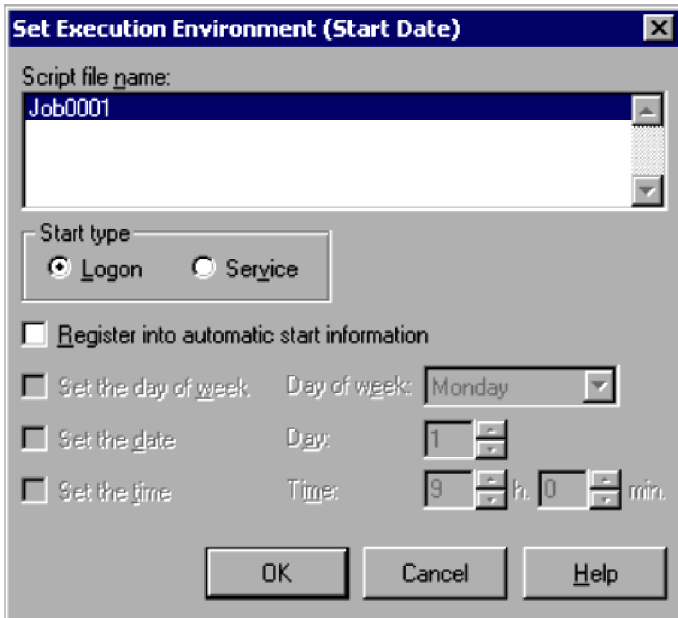
- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the entered setting.
- Click **Cancel** to close the dialog box without updating the environment execution file.

(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.

4.1.14 Set Execution Environment (Start Date) dialog box

The Set Execution Environment (Start Date) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, Start Date**.



Notes

- If you attempt to set up the execution environment (start date) while the JP1/Script service is not running, a dialog box containing the following error message appears:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.
- This setting is not enabled in the Script Launcher service.

(1) Components

In the Set Execution Environment (Start Date) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Start type

Logon or Service

Select whether to start the script at logon or as a service (at power on).

Register into automatic start information

Starts the script automatically at the Script Launcher startup.

Set the day of week[#]

Starts the script on a particular day of the week.

This check box is available only if you select **Register into automatic start information**.

Day of week[#]

Specify a day of the week for weekly execution.

This box is available only if you select **Set the day of week**.

Set the date[#]

Starts the script on a particular day of the month.

This check box is available only if you select **Register into automatic start information**.

Day[#]

Specify a day of the month for monthly execution. The script will not start in a month that does not have the specified day.

This box is available only if you select **Set the date**.

Set the time[#]

Starts the script at a particular time.

This check box is available only if you select **Register into automatic start information**.

Time[#]

Specify the hour and minute for script execution. Enter a value between 00:00 and 23:59. If you specify a time earlier than the Script Launcher start time, the script will run the following day.

This box is available only if you select **Set the time**.

#

This information takes effect only after the JP1/Script service or the Script Launcher is started.

(2) Operations

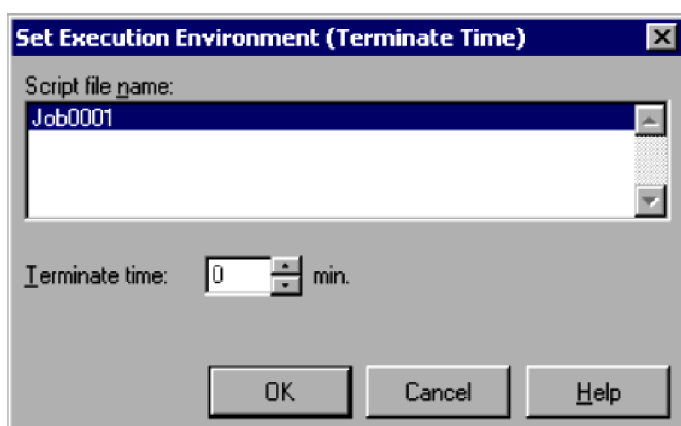
- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the entered settings.
- Click **Cancel** to close the dialog box without updating the environment execution file.

(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.
- If you omit the day of the month and time settings, the script is executed when the JP1/Script service or the Script Launcher starts.

4.1.15 Set Execution Environment (Terminate Time) dialog box

The Set Execution Environment (Terminate Time) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, Terminate Time**.



(1) Components

In the Set Execution Environment (Terminate Time) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Terminate time

Specify the run time in minutes before the script file is forcibly terminated.

Set a number in the range 0 to 1,440. If you set zero, the script file will not be forcibly terminated.

(2) Operations

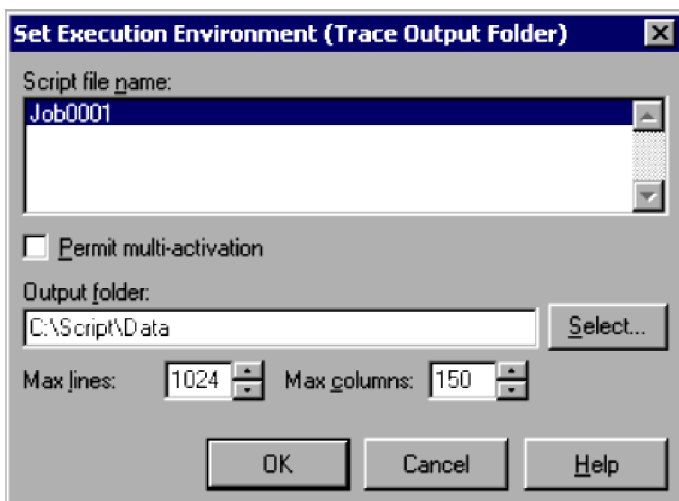
- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the new setting.
- Click **Cancel** to close the dialog box without updating the environment execution file.

(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.
- By entering a termination time, you can enable a process to be automatically terminated if, for some reason, it fails to complete within a particular time.

4.1.16 Set Execution Environment (Trace Output Folder) dialog box

The Set Execution Environment (Trace Output Folder) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, Trace Output Folder**.



(1) Components

In the Set Execution Environment (Trace Output Folder) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Permit multi-activation

Select this check box to allow the script to be activated concurrently from multiple processes.

When multiple execution is enabled, traces are output to the folder specified in **Trace file output folder** in the Options (Multi-activation) dialog box, not the setting in **Output folder**.

There is no limit to the number of concurrent executions. As the number of concurrent executions increases, there is also an increase in the system resources that are required. For this reason, make sure that you pay attention to the number of scripts that are executed concurrently.

Output folder

Specify a folder for the analysis trace file and execution trace file.

If you specify a folder that does not exist, the folder is created when the script file starts.

If you click the **Select** button, the Select Folder dialog box appears.

The entry box is available only if you select **Permit multi-activation**.

Max lines

Specify the maximum number of lines to output from the analysis trace file and execution trace file. Set a number in the range 100 to 9,999.

Max columns

Specify the maximum number of columns to output from the analysis trace file and execution trace file. Set a number in the range 128 to 1,024.

(2) Operations

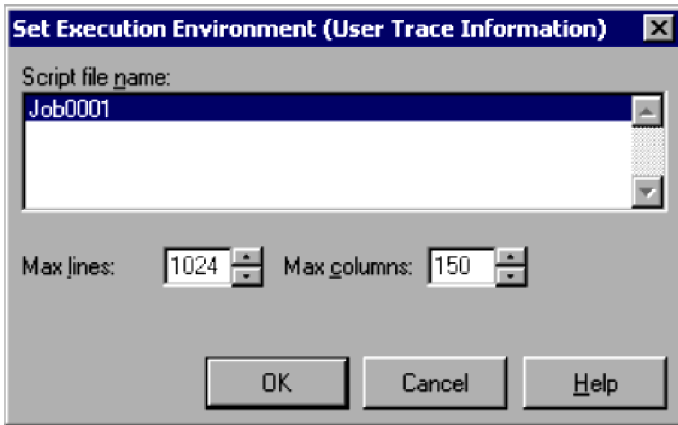
- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the entered setting.
- Click **Cancel** to close the dialog box without updating the environment execution file.

(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.

4.1.17 Set Execution Environment (User Trace Information) dialog box

The Set Execution Environment (User Trace Information) dialog box appears when you select one or more icons in the Script Manager window and choose **File, Set Execution Environment, User Trace Information**.



(1) Components

In the Set Execution Environment (User Trace Information) dialog box, you can set the following items for the script file(s) you selected.

Script file name

Lists the name of the files (icons) that you selected in the Script Manager window. If you selected multiple icons, the file names are sorted in ascending order.

Max lines

Specify the maximum number of lines to output from the user trace file. Set a number in the range 100 to 9,999.

Max columns

Specify the maximum number of columns to output from the user trace file. Set a number in the range 128 to 1,024.

(2) Operations

- Set information about the execution environment as required.
- Click **OK** to update the execution environment file with the entered setting.
- Click **Cancel** to close the dialog box without updating the environment execution file.

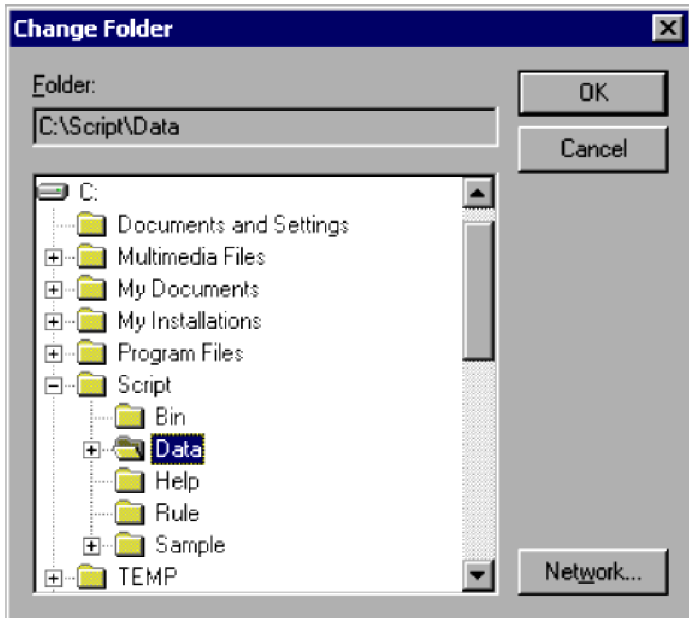
(3) Processing

- When you select multiple icons, items are set by default to the settings for the first file listed in **Script file name**. If you select one of the listed files, the environment settings for that file apply to all the files.

4.1.18 Change Folder dialog box

The Change Folder dialog box appears when you choose **File, Change Folder**.

Change the current folder in the client area to a specified folder. The script files stored in the specified folder will be listed in the client area.



(1) Components

Folder

The name of the folder you select appears at the top. If the folder name exceeds the view area, the last portion of the name is shown.

The drive names and folder names appear in a tree structure at the bottom of the Folder area.

Network

Opens the Assign Network Drives dialog box.

(2) Operations

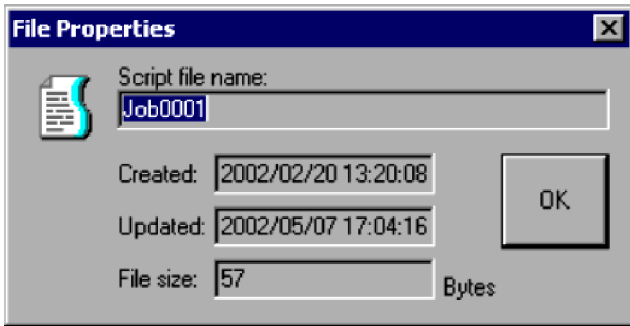
- Select a folder and then click **OK**. The folder changes and information for the new folder appears in the client area.
- Click **Cancel** to close the dialog box without changing the current folder.

(3) Note

You can change the previously changed folder without displaying this dialog box. Simply choose the folder name appearing in the **File** menu.

4.1.19 File Properties dialog box

The File Properties dialog box appears when you choose an icon in the Script Manager window, right-click to display the pop-up menu, and then choose **Properties**.



(1) Components

Script file name

Shows the name of the file (icon) that you selected.

Created

Shows the date and time the file was created.

Updated

Shows the date and time the file was last modified.

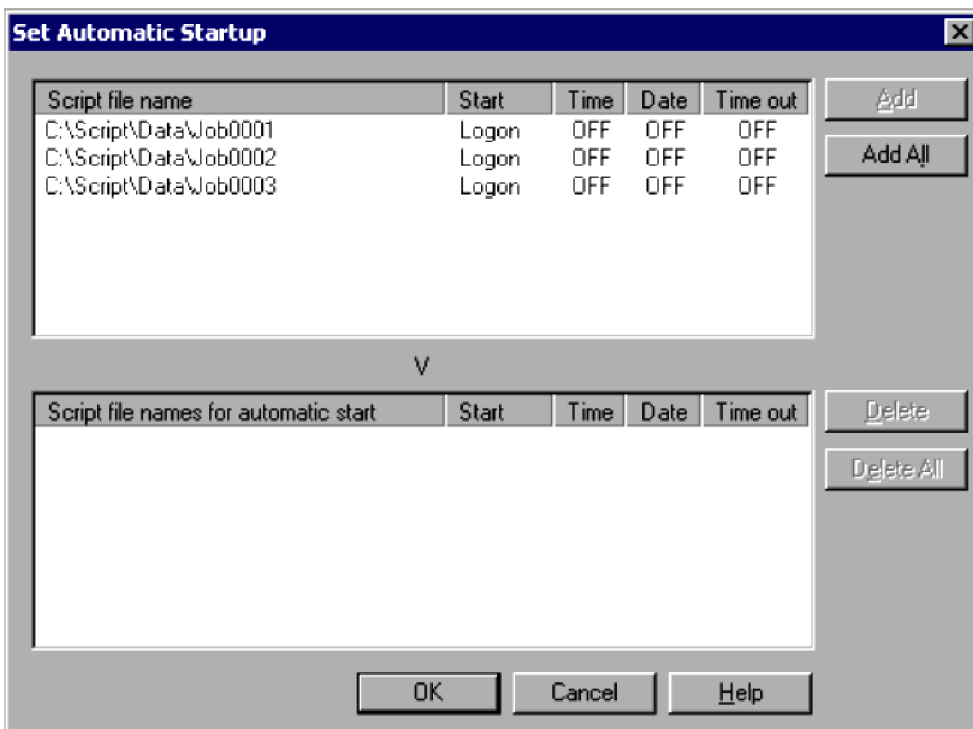
File size

Shows the file size.

4.1.20 Set Automatic Startup dialog box

The Set Automatic Startup dialog box appears when you choose **Tools, Set Automatic Start** in the Script Manager window.

Set the names of the script files to be started automatically by the Script Launcher.



Notes

- If you attempt to set up automatic startup while the JP1/Script service is not running, a dialog box containing the following error message appears:
The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.
- This setting is not enabled in the Script Launcher service.

(1) Components

Script file name

Lists the script files in the current execution folder that have not been registered for automatic startup in the execution environment file.

Script file names for automatic start

Lists the script files that have been registered for automatic startup in the execution environment file.

Start

Shows the start type.

Time

Shows whether or not the file is scheduled to start at a particular time.

ON is displayed if the file is scheduled. **OFF** is specified if the file is not scheduled.

Date

Shows whether or not the file is scheduled to start on a particular day.

ON is displayed if the file is scheduled. **OFF** is specified if the file is not scheduled.

Time out

Shows whether or not a time has been set to forcibly terminate the script.

ON is displayed if the file is scheduled. **OFF** is specified if the file is not scheduled.

(2) Operations

- In the **Script file name** list box, select a script file that you want to start automatically and then click the **Add** button. The selected file is added to the **Script file names for automatic start** list and deleted from the **Script file name** list.
- Click **Add All** to add all the files to the **Script file names for automatic start** list and delete them from the **Script file name** list.
- To remove a script file from the **Script file names for automatic start** list, select the file name and then click **Delete**. The script file is removed from the **Script file names for automatic start** list and added to the **Script file name** list.
- Click **Delete All** to remove all the files from the **Script file names for automatic start** list and add them to the **Script file name** list.
- Click **OK** to output the contents of the **Script file names for automatic start** list box as an automatic start information file.
- Click **Cancel** to close the dialog box without outputting an automatic start information file.

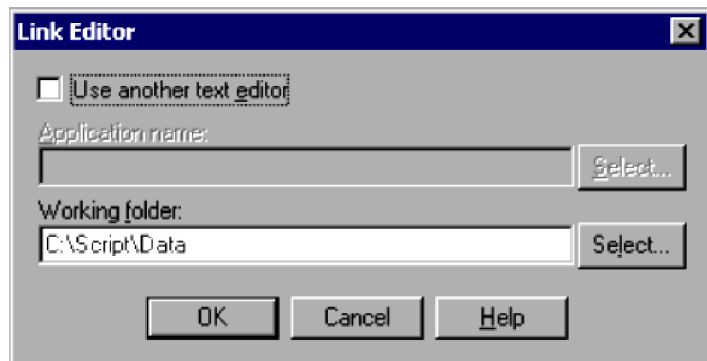
(3) Processing

- When this dialog box first opens, the current contents of the automatic start information file appear in the list box.

- If any script file in the current execution folder does not have an execution environment file, one is created when the dialog box first opens. The file is created irrespective of whether you choose **OK** or **Cancel**.

4.1.21 Link Editor dialog box

The Link Editor dialog box appears when you choose **Tools, Link Editor** in the Script Manager window.



(1) Components

Use another text editor

Choose whether to use an editor other than the default Script Editor.

Application name

Shows the name of the editor to be linked.

You can select an editor by clicking the **Select** button and selecting from the displayed application list.

Working folder

Shows the work folder of the editor to be linked.

You can select a work folder by clicking the **Select** button and selecting from the displayed list.

(2) Operations

- Select an application and then click **OK** to link that application as a script editor.
- Click **Cancel** to close the dialog box without linking an editor.

(3) Processing

- If you do not specify a work folder, the following folder is assumed:

Use another text editor check box selected:

Folder containing the editor

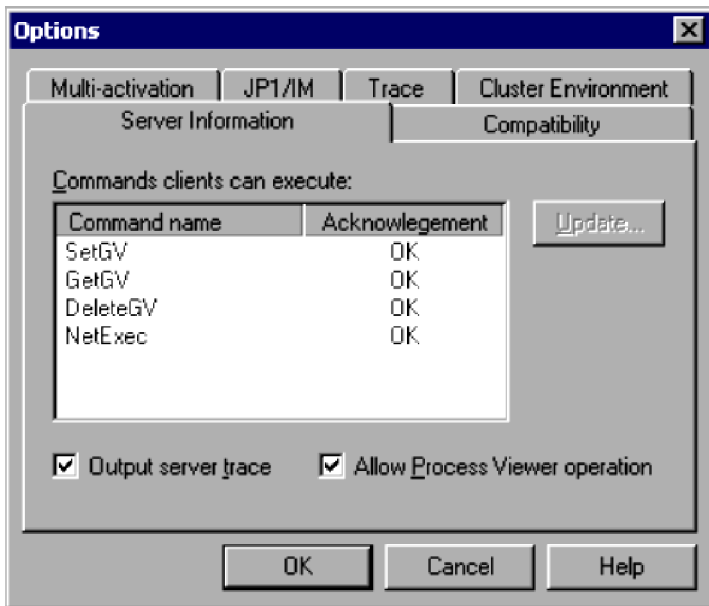
Use another text editor check box cleared:

Folder containing the script file to be edited

4.1.22 Options (Server Information) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **Server Information** tab to display the Options (Server Information) page.



Note

If you attempt to set up the server information options while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Command name

Lists the command names for which you can enable or disable execution from a client.

Acknowledgement

Shows whether clients are allowed to execute a particular command. A circle means execution is permitted; a cross means execution is not permitted.

Update

Displays the Update Information dialog box.

Output server trace

Choose whether to output a trace for commands executed on the server. This box is selected by default.

The output file is fixed as SPTSVTRC.SPY in the DATA folder# in the installation folder.

If you open the Options (Cluster Environment) dialog box from the **Tools** menu and change the folder for management file output, traces will be output to that folder.

#

The server trace is output to the script execution environment folder (*System Drive*\ProgramData\Hitachi\Script\Data).

Allow Process Viewer operation

Choose whether to accept Process Viewer operations performed at a client. This box is selected by default.

(2) Operations

- Click **OK** to apply the settings and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

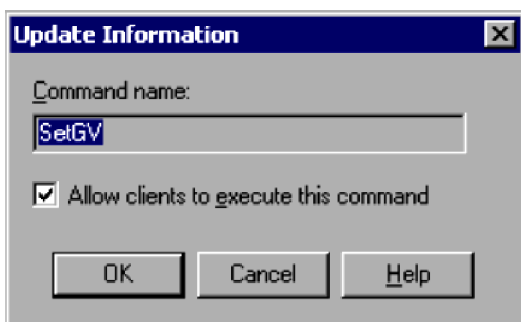
(3) Processing

- Settings other than the **Allow Process Viewer** operation setting are saved in the server environment file (SPTSV.SPS).

4.1.23 Update Information dialog box

The Update Information dialog box appears when you click the **Update** button in the Options dialog box.

Specify whether to allow clients to execute a particular command.



(1) Components

Command name

Shows the command name for which you can change the execution permission.

Allow clients to execute this command

Choose whether to allow clients to execute this command. This check box is selected by default.

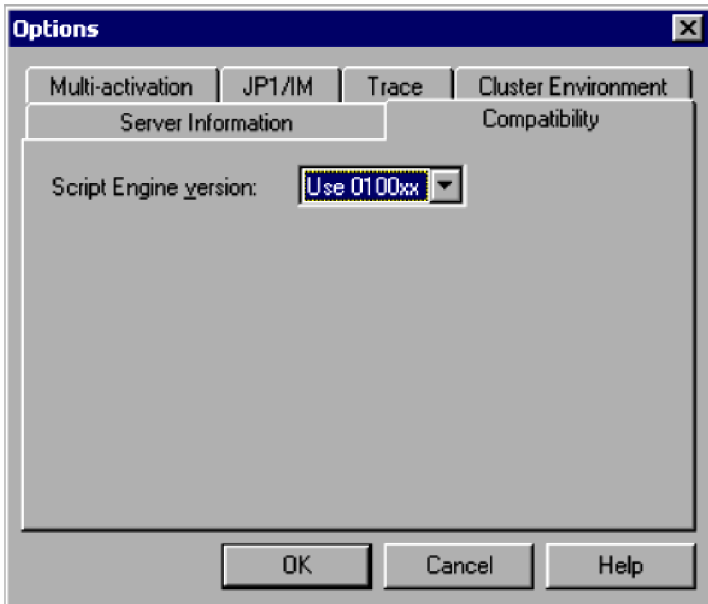
(2) Operations

- Click **OK** to apply the permission setting, close the Update Information dialog box, and return to the Options (Server Information) dialog box.
- Click **Cancel** to close the Update Information dialog box and return to the Options (Server Information) dialog box without changing the permission setting.

4.1.24 Options (Compatibility) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **Compatibility** tab to display the Options (Compatibility) page.



Note

If you attempt to set up the compatibility option while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Script Engine version

Specify the version of Script Engine to be used for executing the script file.

Use 0100xx is set by default.

Important note

The Script Engine version does not need to be the same as the version of the installed JP1/Script program. You can apply the conventions of the version of Script Engine to execute a script file. For details, see [6.1.11 Script coding conventions](#).

(2) Operations

- Click **OK** to apply the setting and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the setting.

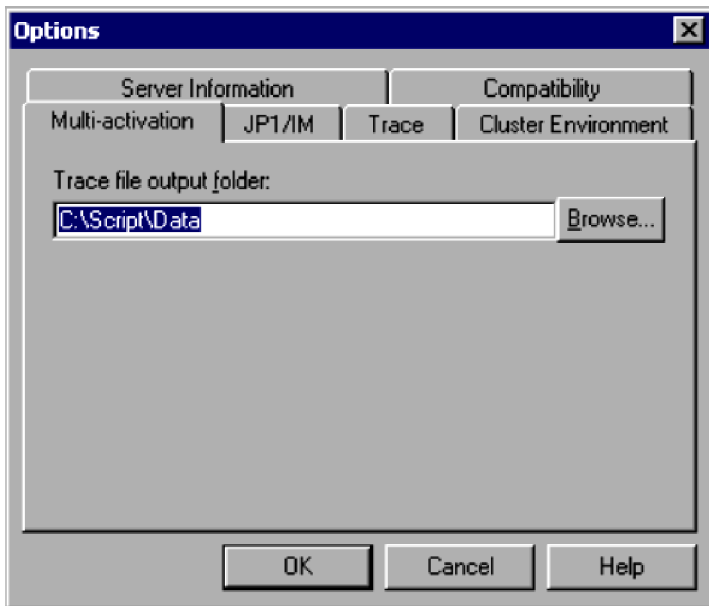
(3) Processing

- JP1/Script uses the version of Script Engine specified here when no Script Engine version is written at the head of the executable script file.

4.1.25 Options (Multi-activation) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **Multi-activation** tab to display the Options (Multi-activation) page.



Note

If you attempt to set up the multi-activation options while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Trace file output folder

Specify a folder for trace file output when the script file is activated concurrently from multiple processes.

You can specify a folder on the local drive only. By default, the folder *JP1/Script-installation-directory\DATA\#* is set here.

The setting here applies only when you select the **Permit multi-activation** check box in either of the following dialog boxes:

- Set Execution Environment (Start Information) dialog box displayed by choosing **File, Set Execution Environment, All Items**
- Set Execution Environment (Trace Output Folder) dialog box displayed by choosing **File, Set Execution Environment, Trace Output Folder**

#

The server trace is output to the script execution environment folder (*System Drive\ProgramData\Hitachi\Script\Data*).

(2) Operations

- Click **OK** to apply the setting and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the setting.

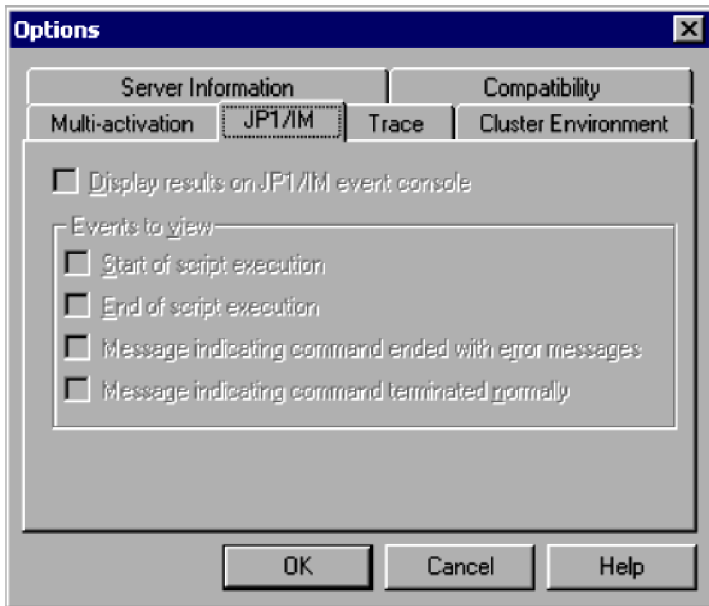
(3) Processing

- The **Trace file output folder** is disabled and the setting here does not apply if you selected the **Change output destination of management file** check box in the Options (Cluster Environment) dialog box (**Tools** menu).

4.1.26 Options (JP1/IM) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **JP1/IM** tab to display the Options (JP1/IM) page.



Note

If you attempt to set up the JP1/IM options while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Display results on JP1/IM event console

Select this check box to display command execution results and other information on the JP1/IM event console.

Events to view

Start of script execution

Display script start events on the event console.

End of script execution

Display script termination events on the event console.

Message indicating command ended with error messages

Display events indicating error termination of a command.

Message indicating command terminated normally

Display events indicating normal termination of a command.

(2) Operations

- Click **OK** to apply the settings and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- You can set the items under **Display results on JP1/IM event console** only if JP1/IM or JP1/Base is installed on your PC. The items are disabled if neither of these programs is installed.
- If JP1/IM or JP1/Base is installed on your PC, the settings in this dialog box apply whether or not the JP1/IM or JP1/Base service is active.
- When you execute a script from Manager (displaying the window) or from the dedicated Editor, script start and termination events are not displayed on the JP1/IM event console even if you select those items in the dialog box.
- You can display execution results on the JP1/IM event console for the following commands:

(a) Basic commands

TextOpen, TextClose, MakeDir, DeleteDir, DeleteFile, Rename, SetFileAttribute, GetFileAttribute, SetFileTime, GetFileTime, GetFileSize, GetVersionInfo, SplitFile, CatFiles, SetStandardFile, ResetStandardFile, SetVolumeLabel, GetVolumeLabel, Copy, Exec, NetExec, WaitForExec, EntryStartUp, and CancelStartUp

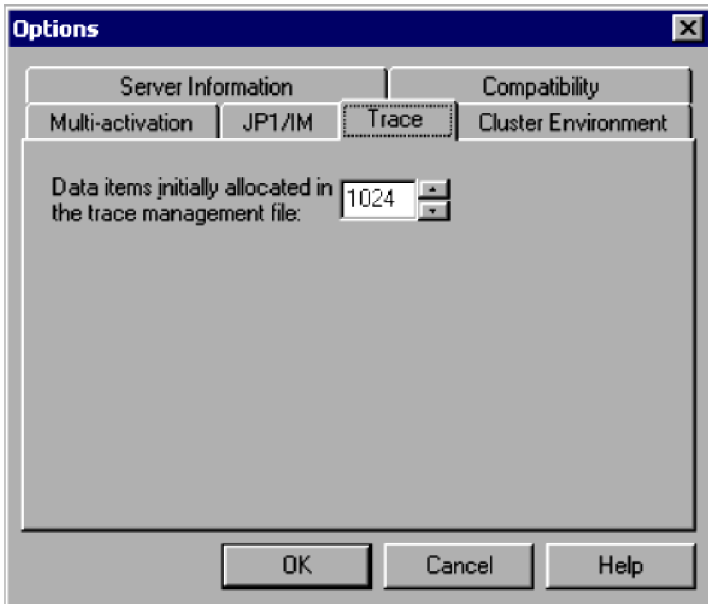
(b) Special commands

ServiceCreate, ServiceDelete, ServiceStart, ServiceStop, ServicePause, ServiceContinue, ServiceChange, ServiceControl, CallDll, MakeGroup, DeleteGroup, MakeShortcut, DeleteShortcut, TerminateProcess, and ExitWindows

4.1.27 Options (Trace) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **Trace** tab to display the Options (Trace) page.



Note

If you attempt to set up the trace options while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Data items initially allocated in the trace management file

Specify the initial data size that can be stored in the trace management file.

Set a value in the range 16 to 32,000. **1,024** is set by default.

(2) Operations

- Click **OK** to apply the setting and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the setting.

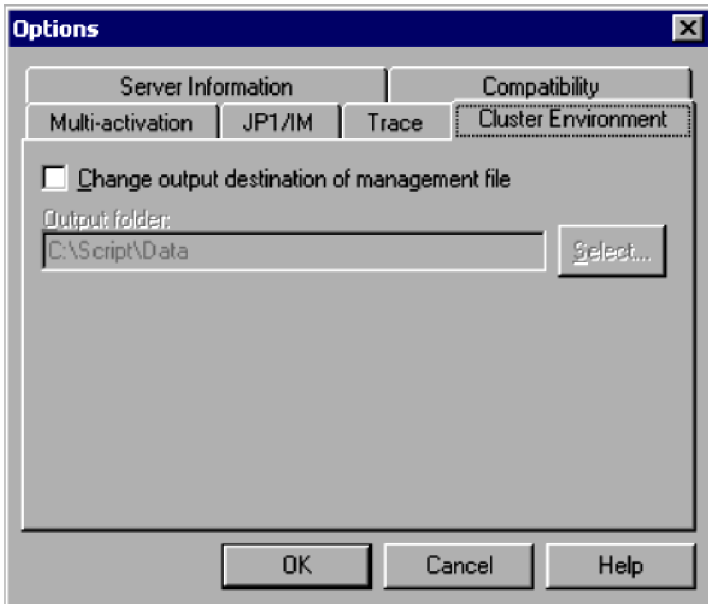
(3) Processing

- Changing the initial data size clears the trace management file (`SPTLOGDB.SPB`). This means that you can no longer view the data previously displayed in the Trace Viewer window. The Script Trace Files Display window may also show incorrect data. If so, close and reopen the Script Trace Files Display window.

4.1.28 Options (Cluster Environment) dialog box

The Options dialog box appears when you choose **Tools, Options** in the Script Manager window.

The Options dialog box has six tabs: **Server Information**, **Compatibility**, **Multi-activation**, **JP1/IM**, **Trace**, and **Cluster Environment**. Click the **Cluster Environment** tab to display the Options (Cluster Environment) page.



Note

If you attempt to set up the cluster environment options while the JP1/Script service is not running, a dialog box containing the following error message appears:

The operation could not be performed because the JP1/Script service is not running. Launch the JP1/Script service, and then re-execute the operation.

(1) Components

Change output destination of management file

Changes the default folder for management file output (normally the DATA folder[#] in the JP1/Script installation folder).

Management files refer to the following files handled by JP1/Script: automatic start information file, server environment file, server trace file, trace management file, and global variables file.

Output Folder

If you selected the **Change output destination of management file** check box, specify the folder for management file output.

Specify a folder in a shared directory in the clustering environment. You must specify a folder other than the DATA folder[#] in the JP1/Script installation folder.

The server trace is output to the script execution environment folder (*System Drive*\ProgramData\Hitachi\Script\Data).

(2) Operations

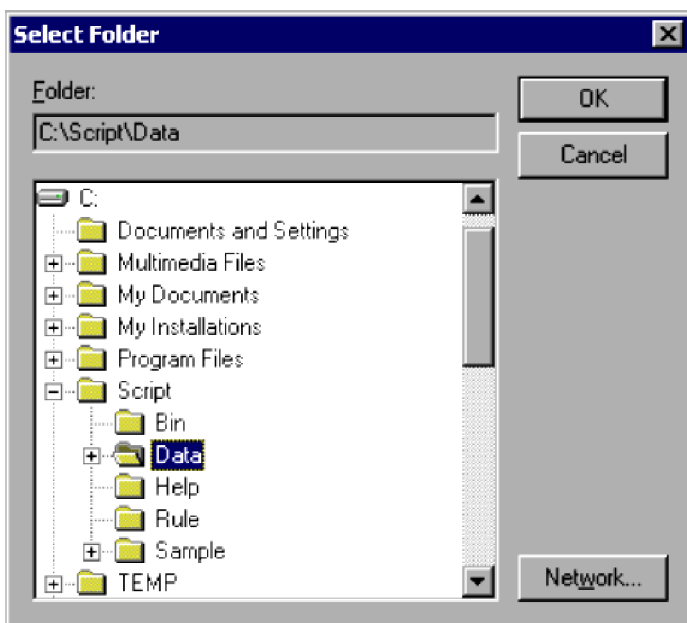
- If you click **OK**, a dialog box asks for confirmation that you want to change the output folder and restart. Choose the **Restart** button to apply the setting, close the Options dialog box, and restart the system.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- JP1/Script can be used in a clustering environment, but it is not possible to resume execution of scripts that were running when the failover occurred. However, the execution environment can be inherited by the standby system if you change the default folder for management file output to a folder on the shared disk, using this dialog box. For details, see *2.3 Environment setup in a cluster system environment*.
- You must restart the system to apply the new setting in this dialog box.
If any management files were already in use before you changed the output folder, those files remain in the previous folder and their contents become invalid.
If any management files already exist in the new output folder, JP1/Script references those files and recreates any that are missing.
- Multiple computers cannot concurrently reference a file that is output to the output folder.
- If you select the **Change output destination of management file** check box, the trace files produced when the script file is activated concurrently from multiple processes are output to the Trace folder in the specified output folder. The folder specified in **Trace file output folder** in the Options (Multi-activation) dialog box (**Tools** menu) does not apply.
- When you set a new output folder, the contents of the trace management file (SPTLOGDB . SPB) may not match the contents of the trace files. As a result, the contents displayed in the Trace Viewer window (file size and date modified) may be incorrect.

4.1.29 Select Folder dialog box

The Select Folder dialog box appears when you choose the **Select** button to select a folder in a dialog box.



(1) Components

Folder

The name of the folder you select appears at the top. If the folder name exceeds the view area, the last portion of the name is shown.

The drive names and folder names appear in a tree structure at the bottom of the **Folder** area.

Network

Displays the Assign Network Drives dialog box.

(2) Operations

- Click **OK** to apply information for the selected folder.
- Click **Cancel** to close the dialog box without applying the selected folder information.

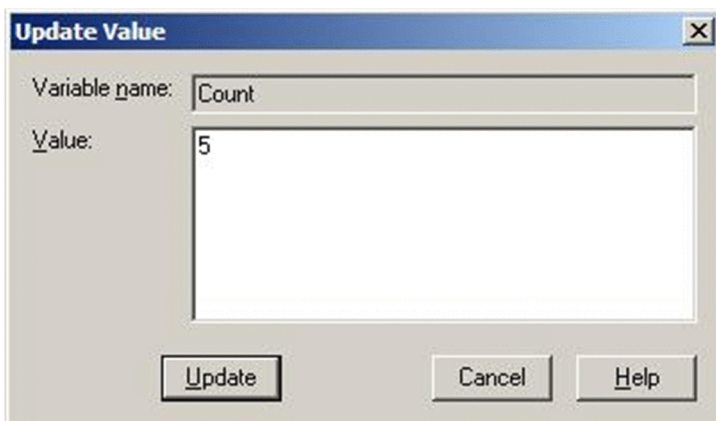
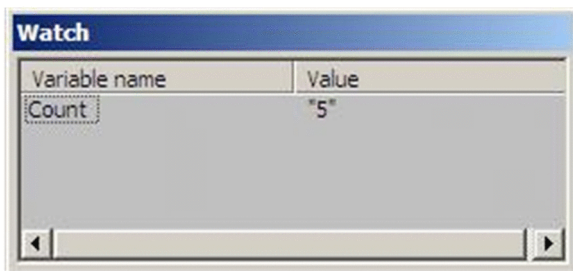
4.2 Script Editor dialog boxes

The dialog boxes displayed when you use the Script Editor window are listed below. Detailed descriptions of each dialog box and its components are given after this list.

- Update Value dialog box
- Options (Format) dialog box
- Options (Colors) dialog box
- Options (Compatibility) dialog box
- Set dialog box
- Search dialog box
- Set File Name dialog box
- Set File Version dialog box
- Command Line Parameter Settings dialog box
- Add Variable dialog box

4.2.1 Update Value dialog box

The Update Value dialog box appears when you double-click a variable name in the Watch window.



(1) Components

Variable name

Shows the name of the variable to be updated.

Value

Enter a new value of up to 1,024 bytes. This field is disabled if the variable name is invalid.

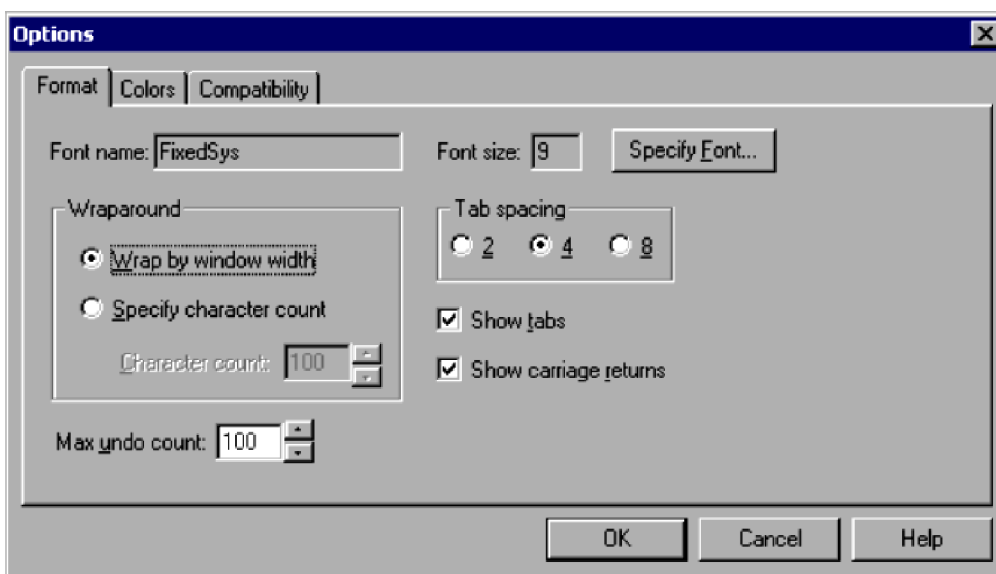
(2) Operations

- Click **Update** to update the value of the specified variable and close the Update Value dialog box.
- Click **Cancel** to close the dialog box without updating the variable value.

4.2.2 Options (Format) dialog box

The Options dialog box appears when you choose **Edit, Options** in the Script Editor window.

The Options dialog box has three tabs: **Format**, **Colors**, and **Compatibility**. Click the **Format** tab to display the Options (Format) page.



(1) Components

Font name

Shows the font name. To change the font, click the **Specify Font** button.

FixedSys is set by default.

Font size

Shows the font size. To change the font size, click the **Specify Font** button.

14 is set by default.

Wraparound

Select a text wrap method.

Wrap by window width is selected by default.

Wrap by window width

Wrap text to the window width.

Specify character count

Wrap text at a fixed number of characters.

Character count

Set the number of characters at which to wrap text.

This box is available only when you select **Specify character count**.

Set a value in the range 20 to 512.

Tab spacing

Set the number of tabs.

4 is set by default.

Show tabs

Choose whether to show or hide tabs. This check box is selected by default.

Show carriage returns

Choose whether to show or hide carriage returns. This check box is selected by default.

Max undo count

Specify how many times the user can choose **Edit**, **Undo**.

Set a value in the range 10 to 999. **100** is set by default.

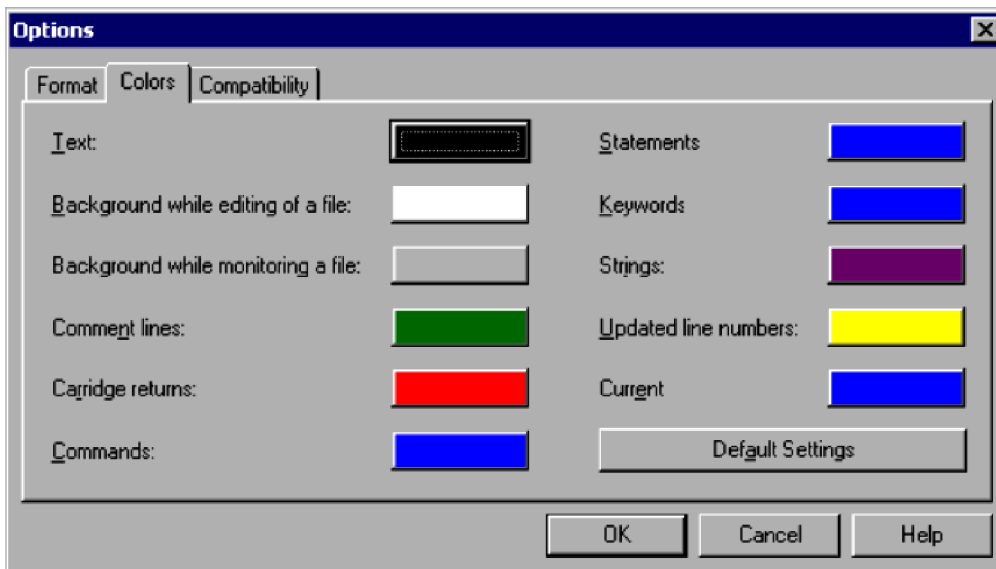
(2) Operations

- Click **OK** to apply the format settings and close the Options dialog box.
- Click **Cancel** to close the dialog box without changing the existing format settings.

4.2.3 Options (Colors) dialog box

The Options dialog box appears when you choose **Edit, Options** in the Script Editor window.

The Options dialog box has three tabs: **Format**, **Colors**, and **Compatibility**. Click the **Colors** tab to display the Options (Colors) page.



(1) Components

Text

Set the text color.

The system color is set by default.

Background while editing of a file

Set the background color for edit mode.

The system color is set by default.

Background while monitoring a file

Set the background color for monitoring mode.

Gray is set by default.

Comment lines

Set the color of comment lines.

Green is set by default.

Carriage returns

Set the color of line returns.

Red is set by default.

Commands

Set the color of commands.

Blue is set by default.

Statements

Set the color of statements.

Blue is set by default.

Keywords

Set the color of keywords.

Blue is set by default.

Strings

Set the color of character strings.

Violet is set by default.

Updated line numbers

Set the color of the line numbers of modified lines.

Yellow is set by default.

Current

Set the color of the cursor that indicates the current position.

Blue is set by default.

Default Settings

Resets all colors to the defaults.

(2) Operations

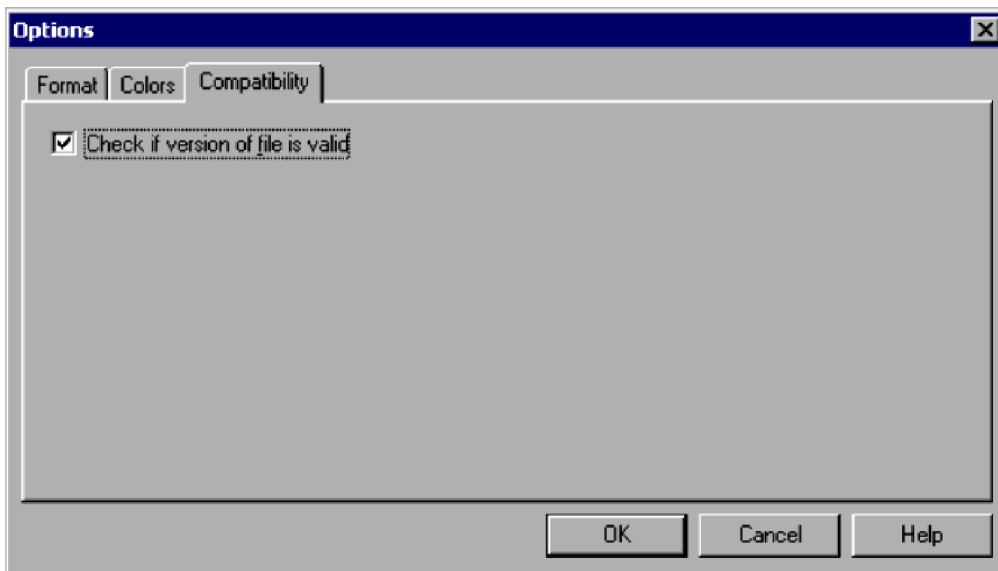
- Click the button for any item (except **Default Settings**) to display the Set Color dialog box and set a color for that item. In the Set Color dialog box, you can choose the **Create Color** button to display the Create Color dialog box and create a color.

- Click **OK** to set the specified colors and close the Options dialog box.
- Click **Cancel** to close the dialog box without changing the existing colors.

4.2.4 Options (Compatibility) dialog box

The Options dialog box appears when you choose **Edit, Options** in the Script Editor window.

The Options dialog box has three tabs: **Format**, **Colors**, and **Compatibility**. Click the **Compatibility** tab to display the Options (Compatibility) page.



(1) Components

Check if version of file is valid

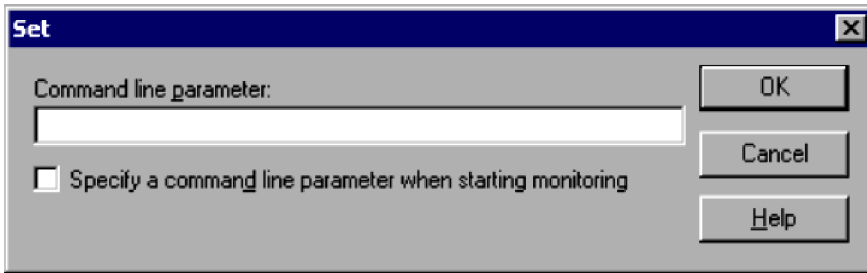
Choose whether to check if the file version is specified at the head of the file when saving a script file.

(2) Operations

- Click **OK** to apply the setting and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the setting.

4.2.5 Set dialog box

The Set dialog box appears when you choose **Monitoring, Set** in the Script Editor window.



(1) Components

Command line parameter

Specify the parameters required for monitoring.

Specify a command line parameter when starting monitoring

Select this check box to display the Command Line Parameter Settings dialog box and specify the command line parameters when you begin monitoring.

(2) Operations

- Click **OK** to apply the settings and close the Options dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- You can set up to 1,024 characters as command line parameters.

If you set parameters both in this dialog box and in the command line in the script execution environment file, all the specified values will apply when you begin monitoring.

The values you set in the dialog box are applied first, then the values set in the execution environment file. If different values are set for the same item (for example, `/SPXLV (3)` in the dialog box, but `/SPXLV (0)` in the file), the dialog box setting takes precedence.

Example:

Below, location variable %1 is ABC, and location variable %2 is 123. Parameter `/SPXLV (3)` takes precedence, and `/NOEVLOG` is valid.

Command line specified in the dialog box:

```
/SPXLV (3) ABC
```

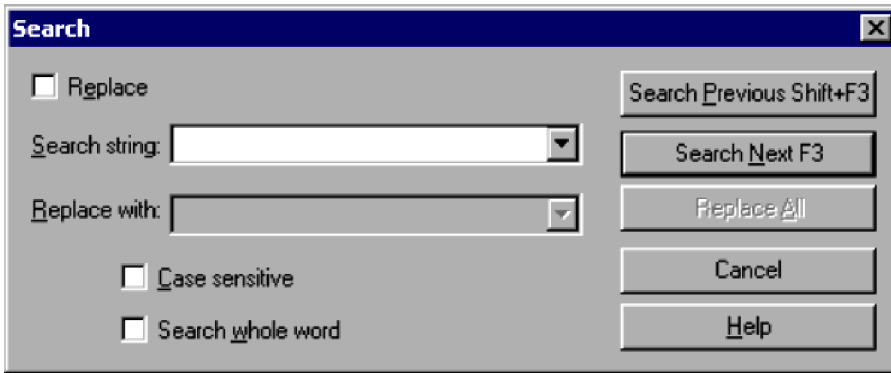
Command line specified in the execution environment file:

```
/SPXLV (0) 123 /NOEVLOG
```

- The settings in the Set dialog box are saved to a monitoring information file when you quit the editor.

4.2.6 Search dialog box

The Search dialog box appears when you choose **Search, Find** in the Script Editor window. It also appears when you choose **Find Next** or **Find Previous** from the **Search** menu.



(1) Components

Replace

Select this check box to replace the search string.

Search string

Enter a search string.

Replace with

Enter text to replace the search string. This box is available only if you select **Replace**.

Case sensitive

Distinguish between uppercase and lowercase when searching for the string. This check box is unselected by default.

Search whole word

Search for whole words only. This check box is unselected by default.

(2) Operations

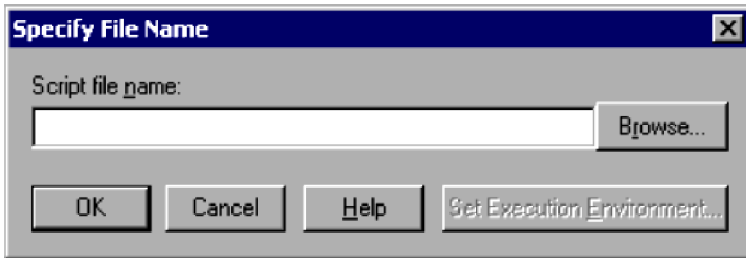
- Click the **Search Previous** button to search upward from the current position.
- Click the **Search Next** button to search downward from the current position.
- Click **Replace All** to replace all the matching search strings in the script file with the text specified in **Replace with**.
- Click **Cancel** to close the dialog box.

(3) Processing

- The **Search string** and **Replace with** drop-down lists contain the 10 most recent strings.
- A beep sounds if the search string is not found.

4.2.7 Set File Name dialog box

The Set File Name dialog box appears when you attempt to execute or monitor an unnamed script file in the Script Editor window.



(1) Components

Script file name

Enter the script file name.

(2) Operations

- Click **Browse** to display the Open File dialog box.
- Click **OK** to save the script file under the specified file name and begin execution or monitoring.
- Click **Cancel** to close the dialog box without naming the file.
- Click **Set Execution Environment** to open the Set Execution Environment dialog box.

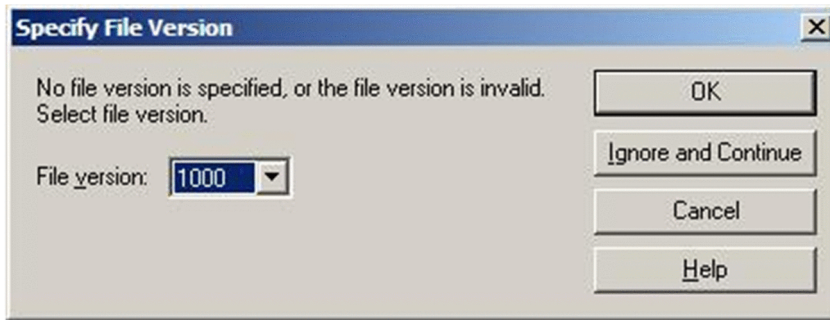
4.2.8 Set File Version dialog box

You can use the Set File Version dialog box for a script file that satisfies both of the following conditions:

- **Check if version of file is valid** check box selected in the Options (Compatibility) dialog box
- File version not written at the head of the script file

The Set File Version dialog box appears when you perform one of the following operations in the Script Editor window:

- Choose **File, Save**.
- Choose **File, Save As**.
- Display a different script file.
- Choose **Monitoring, Syntax Check**.
- Choose **Monitoring, Execute Monitoring, Execution**.
- Choose **Monitoring, Execute Monitoring, Step Execution**.
- Choose **Monitoring, Execute Monitoring, Consecutive Step Execution**.
- Choose **Monitoring, Execution**.
- Quit Editor.



(1) Components

File version

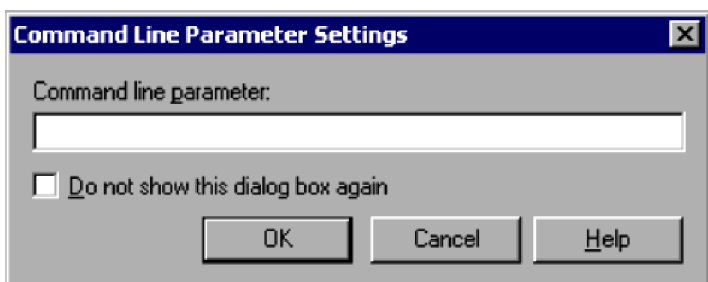
Specify the file version to be written at the head of the script file.

(2) Operations

- Click **OK** to write the specified version at the head of the script file and save the file.
- Click **Ignore and Continue** to save the script file without writing a version specification, and then continue whichever of the above operations you were performing.
- Click **Cancel** to cancel whichever of the above operations you were performing. The version you specified is not written at the head of the script file and the script file is not saved.

4.2.9 Command Line Parameter Settings dialog box

The Command Line Parameter Settings dialog box appears when you perform monitoring, step execution, or consecutive step execution, having selected the **Specify a command line parameter when starting monitoring** check box in the Set dialog box displayed when you choose **Monitoring, Set** in the Script Editor window.



(1) Components

Command line parameter

Specify the parameters required for monitoring.

Do not show this dialog box again

Select this check box if you do not want to display this dialog box the next time you begin monitoring.

(2) Operations

- Click **OK** to apply the settings, close the Command Line Parameter Settings dialog box, and begin monitoring.

- Click **Cancel** to close the dialog box and begin monitoring without setting these parameters.

(3) Processing

- You can set up to 1,024 characters as command line parameters.

If you set parameters both in this dialog box and in the command line in the script execution environment file, all the specified values will apply when you begin monitoring.

The values you set in the dialog box are applied first, then the values set in the execution environment file. If different values are set for the same item (for example, `/SPXLV (3)` in the dialog box, but `/SPXLV (0)` in the file), the dialog box setting takes precedence.

Example:

Below, location variable %1 is ABC, and location variable %2 is 123. Parameter `/SPXLV (3)` takes precedence, and `/NOEVLOG` is valid.

Command line specified in the dialog box:

```
/SPXLV (3) ABC
```

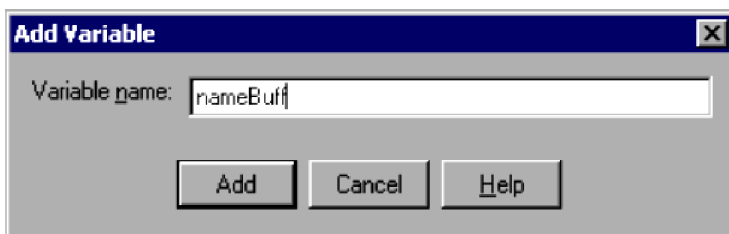
Command line specified in the execution environment file:

```
/SPXLV (0) 123 /NOEVLOG
```

- The settings in the Command Line Parameter Settings dialog box are saved to a monitoring information file when you quit the editor.

4.2.10 Add Variable dialog box

The Add Variable dialog box appears when you choose **Monitoring, Add to Watch Window** in the Script Editor window.



(1) Components

Variable name

Enter the name of the variable to be added.

(2) Operations

- Click **Add** to add the specified variable name to the Watch window and close the dialog box.
- Click **Cancel** to close the dialog box without adding the variable.

(3) Processing

- The **Monitoring, Add to Watch Window** command is available only in monitoring mode.
- Specifying the same variable more than once, or specifying a non-existent variable, does not result in an error.

- You can write up to 99 characters as a variable name. Names longer than 99 characters are truncated.

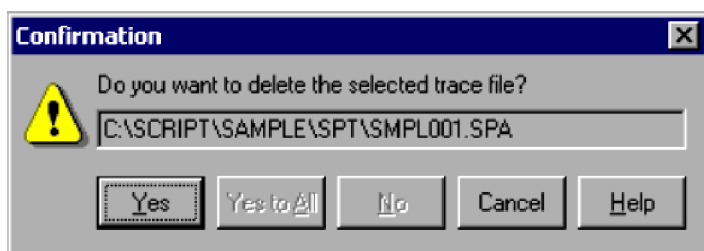
4.3 Script Trace Viewer dialog boxes

The dialog boxes displayed when you use the Script Trace Viewer window are listed below. Detailed descriptions of each dialog box and its components are given after this list.

- Confirmation (delete trace file) dialog box
- Confirmation (clear trace file) dialog box
- Select Computer dialog box
- Search dialog box

4.3.1 Confirmation (delete trace file) dialog box

The Confirmation (delete trace file) dialog box appears when you select one or more trace files and choose **File, Delete Trace File**.



(1) Components

Do you want to delete the selected trace file?

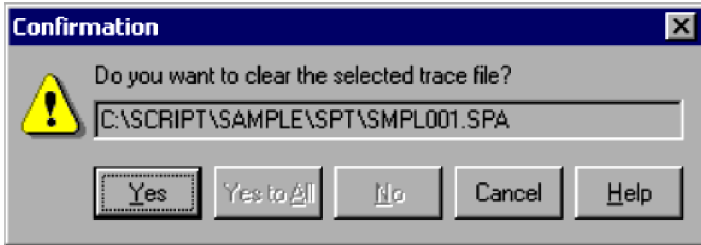
Shows the name of a trace file you selected.

(2) Operations

- Click **Yes** to delete the displayed trace file. If you selected more than one file, the next file is displayed.
- Click **Yes to All** to delete all the trace files you selected.
- Click **No** if you do not want to delete the displayed trace file. If you selected more than one file, the next file is displayed.
- Click **Cancel** to stop deleting the trace file(s) and close the dialog box.

4.3.2 Confirmation (clear trace file) dialog box

The Confirmation (clear trace file) dialog box appears when you select one or more trace files and choose **Edit, Clear Trace File**.



(1) Components

Do you want to clear the selected trace file?

Shows the name of a trace file you selected.

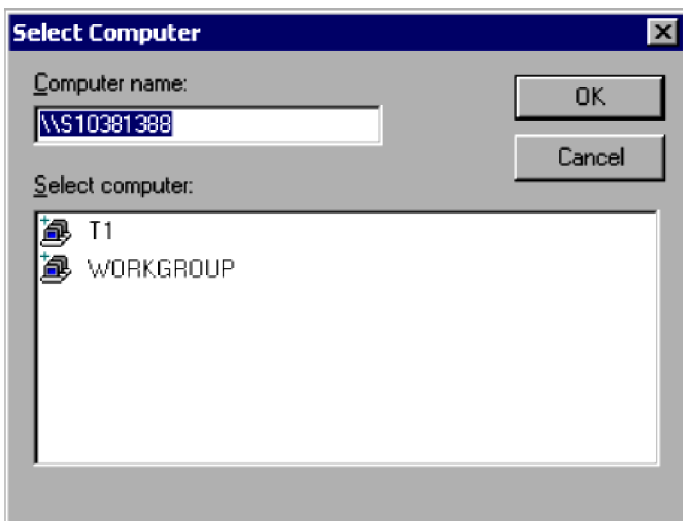
(2) Operations

- Click **Yes** to clear the displayed trace file. If you selected more than one file, the next file is displayed.
- Click **Yes to All** to clear all the trace files you selected.
- Click **No** if you do not want to clear the displayed trace file. If you selected more than one file, the next file is displayed.
- Click **Cancel** to stop clearing the trace file(s) and close the dialog box.

4.3.3 Select Computer dialog box

The Select Computer dialog box appears when you choose **Options, Select Computer**.

Change the computer connection.



(1) Components

Computer name

Specify the name of the computer to connect to (maximum 15 characters).

Select computer

Lists the computers connected to the local computer.

(2) Operations

- Select a computer in the list and then click **OK** to connect to that computer.
- Click **Cancel** to close the dialog box without changing the computer connection.

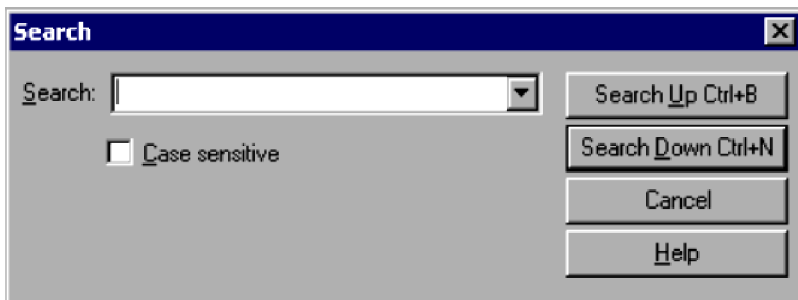
(3) Processing

- The computer name can contain up to 15 characters.
- The client area shows nothing if the specified computer cannot be connected.

4.3.4 Search dialog box

The Search dialog box appears when you choose **Search, Find** in the Script Trace Files Display window.

Search for specified text in the active trace file.



(1) Components

Search

Enter a search string.

Case sensitive

Select this check box to distinguish between uppercase and lowercase as a search condition.

(2) Operations

- Enter a search string, then click the **Search Up** button to search upward from the current position. Click the **Search Down** button to search downward from the current position.
- Click **Cancel** to close the dialog box without conducting the search.

(3) Processing

- If you select text in the active trace file and then choose the **Find** command, the selected text is assumed to be the search string.

4.4 Script Menu Editor dialog boxes

The dialog boxes displayed when you use the Script Menu Editor window are listed below. Detailed descriptions of each dialog box and its components are given after this list.

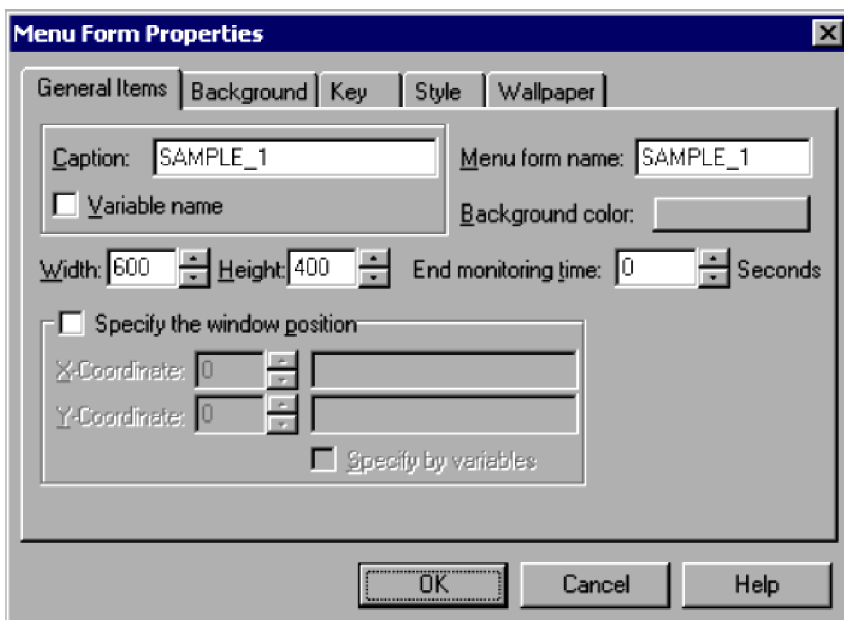
- Menu Form Properties (General Items) dialog box
- Menu Form Properties (Background) dialog box
- Menu Form Properties (Key) dialog box
- Menu Form Properties (Style) dialog box
- Menu Form Properties (Wallpaper) dialog box
- Static Properties (General Items) dialog box
- Static Properties (Common Items) dialog box
- Static Properties (Background) dialog box
- Static Properties (Style) dialog box
- Button Properties (General Items) dialog box
- Button Properties (Common Items) dialog box
- Button Properties (Background) dialog box
- Button Properties (Key) dialog box
- Button Properties (Style) dialog box
- Browse Button Properties (General Items) dialog box
- Browse Button Properties (Common Items) dialog box
- Browse Button Properties (Background) dialog box
- Browse Button Properties (Key) dialog box
- Browse Button Properties (Style) dialog box
- Edit Control Properties (General Items) dialog box
- Edit Control Properties (Common Items) dialog box
- Edit Control Properties (Key) dialog box
- Edit Control Properties (Style) dialog box
- Line Properties (General Items) dialog box
- Function Key Properties (General Items) dialog box
- Function Key Properties (Common Items) dialog box
- Function Key Properties (Style) dialog box
- List Control Properties (General Items) dialog box
- List Control Properties (Common Items) dialog box
- List Control Properties (Key) dialog box
- List Control Properties (Style) dialog box
- Combo Box Properties (General Items) dialog box
- Combo Box Properties (Common Items) dialog box
- Combo Box Properties (Key) dialog box

- Combo Box Properties (Style) dialog box
- Command Properties dialog box
- Set Command Properties dialog box
- Set Menu Form Name dialog box
- Change as Batch dialog box
- Set Grid dialog box
- Set Tab Order dialog box
- Print Information of Menu Forms dialog box

4.4.1 Menu Form Properties (General Items) dialog box

The Menu Form Properties dialog box appears when you select and double-click a menu form in the menu form view of the Script Menu Editor window.

The Menu Form Properties dialog box has five tabs: **General Items**, **Background**, **Key**, **Style**, and **Wallpaper**. Click the **General Items** tab to display the Menu Form Properties (General Items) page.



(1) Components

In the Menu Form Properties (General Items) dialog box, you can set the following items for the menu form you selected.

Caption

Enter a caption for the menu form. You can specify character strings or a variable name that stores character strings. You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable in **Caption**.

Menu form name

Type the name of the menu form, complying with the variable naming conventions.

End monitoring time

Specify a timeout for the menu form.

Set a value in the range 0 to 86,400 (seconds). If you set zero, the menu form timeout will not be monitored.

Width

Specify in pixels the width of the menu form.

Set a value in the range 115 to 9,999. **600** is set by default.

Height

Specify in pixels the height of the menu form.

Set a value in the range 22 to 9,999. **400** is set by default.

Background color

Shows the background color of the menu form as specified in the Set Color dialog box.

Gray is set by default.

Specify the window position

Select this check box to set the position of the menu form when activated.

X Coordinate and Y Coordinate

Specify the X and Y coordinates of the display position when the menu form is activated. Set the values in pixels or as variables that store the values.

These boxes are available only if you select the **Specify the window position** check box. **Zero** is set in both coordinates by default.

Specify by variables

Select this check box if you want to specify variable names for **X Coordinate** and **Y Coordinate**.

This check box is available only if you select **Specify the window position**.

If you select this check box, you need to specify variable names for both **X Coordinate** and **Y Coordinate**.

You cannot specify an array variable for a variable name.

(2) Operations

- Click **OK** to apply the settings and close the Menu Form Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

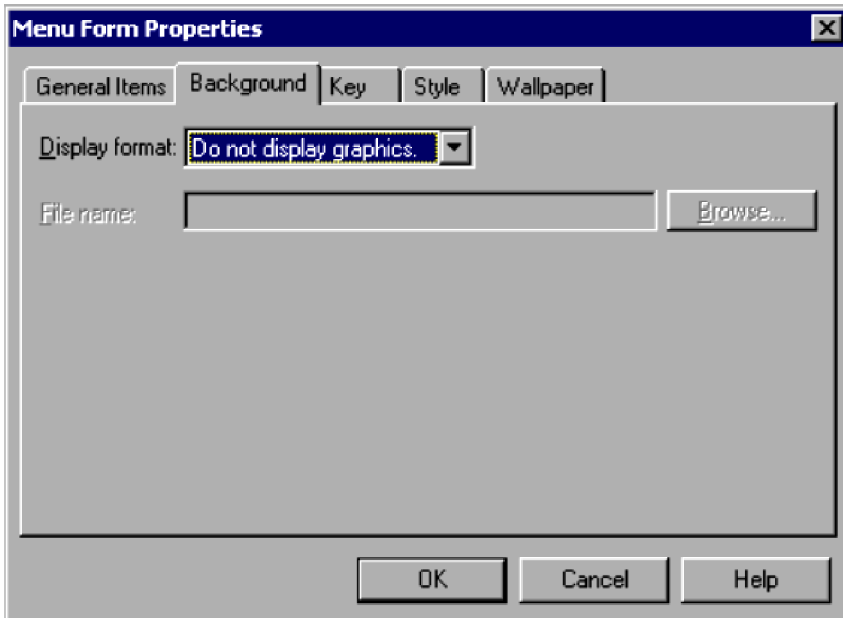
(3) Processing

- When the **Specify the window position** check box is unselected, the activated menu form appears in the center of the screen.

4.4.2 Menu Form Properties (Background) dialog box

The Menu Form Properties dialog box appears when you select and double-click a menu form in the menu form view of the Script Menu Editor window.

The Menu Form Properties dialog box has five tabs: **General Items**, **Background**, **Key**, **Style**, and **Wallpaper**. Click the **Background** tab to display the Menu Form Properties (Background) page.



(1) Components

In the Menu Form Properties (Background) dialog box, you can set the following items for the background of the menu form you selected.

Display format

From the drop-down list, select how to display the background image.

Do not display graphics is set by default.

File name

Enter the file name of the image to display in the background.

This field is disabled when **Do not display graphics** is selected in **Display format**.

Browse

Click to select an image file.

This button is disabled when **Do not display graphics** is selected in **Display format**.

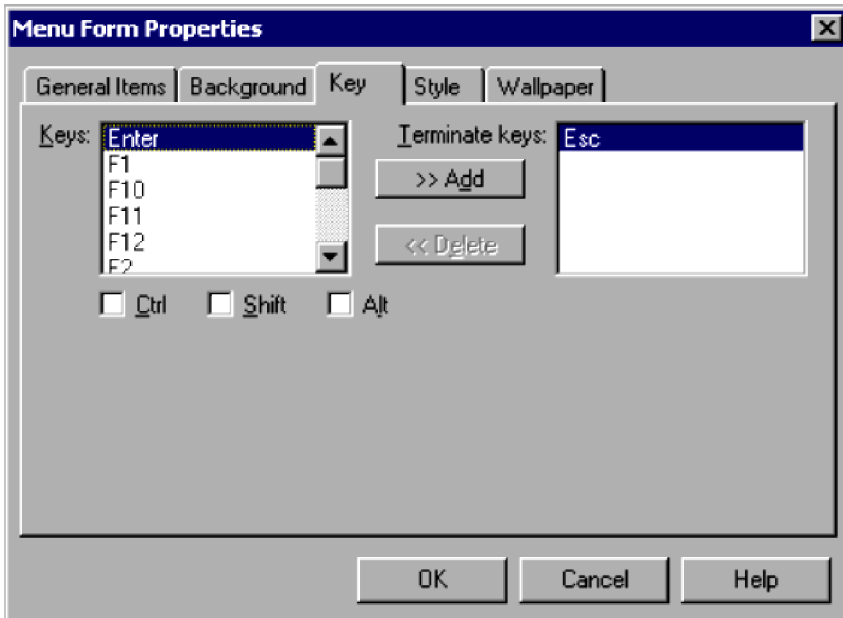
(2) Operations

- Click **OK** to apply the settings and close the Menu Form Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.3 Menu Form Properties (Key) dialog box

The Menu Form Properties dialog box appears when you select and double-click a menu form in the menu form view of the Script Menu Editor window.

The Menu Form Properties dialog box has five tabs: **General Items**, **Background**, **Key**, **Style**, and **Wallpaper**. Click the **Key** tab to display the Menu Form Properties (Key) page.



(1) Components

In the Menu Form Properties (Key) dialog box, you can set the following items for the keys of the menu form you selected.

Keys

Select a key to set as an exit key. You can select **Esc**, **Enter**, or **F1** to **F12**.

Ctrl

Sets the **Ctrl** key as a modifier key.

Shift

Sets the **Shift** key as a modifier key.

Alt

Sets the **Alt** key as a modifier key.

>> Add

Adds a selected key as an exit key.

<< Delete

Deletes a key selected in the **Terminate keys** list.

Terminate keys

Select an exit key to delete. **Esc** is set by default.

(2) Operations

- Click **OK** to apply the settings and close the Menu Form Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- If **Esc** is selected for **Terminate keys**, the menu form will be closed without checking the entered data or assigning a value to a variable. If a value other than **Esc** is selected for **Terminate keys**, the menu form will be closed after the entered data is checked and the values are assigned to the variables.

- When function keys are included in the menu form, **F1** to **F12** cannot be selected as exit keys and do not appear in the **Keys** list.

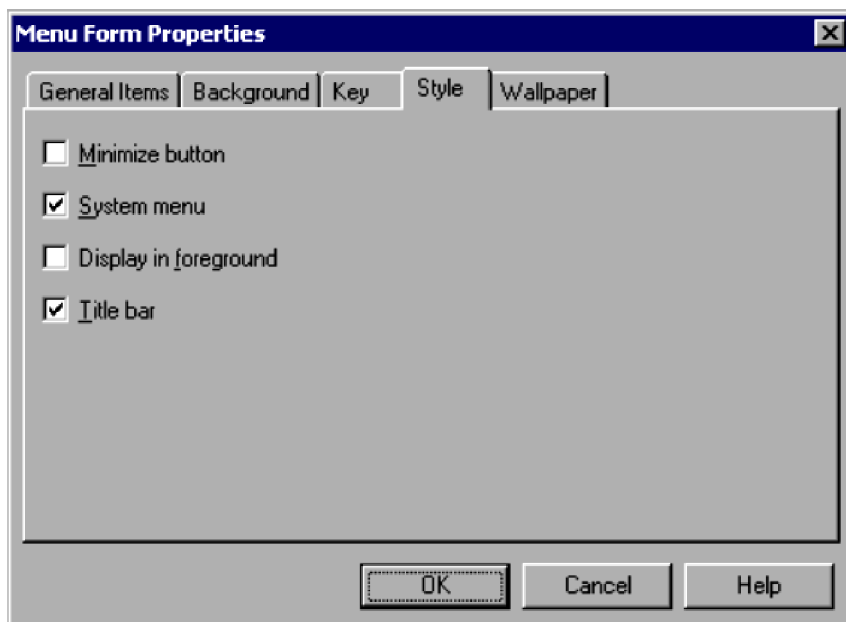
In the following cases, **F1** to **F12** are invalid if set as exit keys, and a warning message appears when the menu form opens:

- The user defines a function key, but **F1** to **F12** are already set in Terminate keys.
- A function key has been defined, and **F1** to **F12** are set in Terminate keys, in a menu form created using a version of JP1/Script earlier than version 06-71.

4.4.4 Menu Form Properties (Style) dialog box

The Menu Form Properties dialog box appears when you select and double-click a menu form in the menu form view of the Script Menu Editor window.

The Menu Form Properties dialog box has five tabs: **General Items**, **Background**, **Key**, **Style**, and **Wallpaper**. Click the **Style** tab to display the Menu Form Properties (Style) page.



(1) Components

In the Menu Form Properties (Style) dialog box, you can set the following items for the menu form you selected.

Minimize button

Attaches a minimize button to the menu form. This check box is available only when **Title bar** is selected.

System menu

Sets a system menu on the menu form. This check box is available only when **Title bar** is selected. **System menu** is selected by default.

Display in foreground

Displays the menu form in the foreground.

Title bar

Attaches a title bar to the menu form. This check box is selected by default.

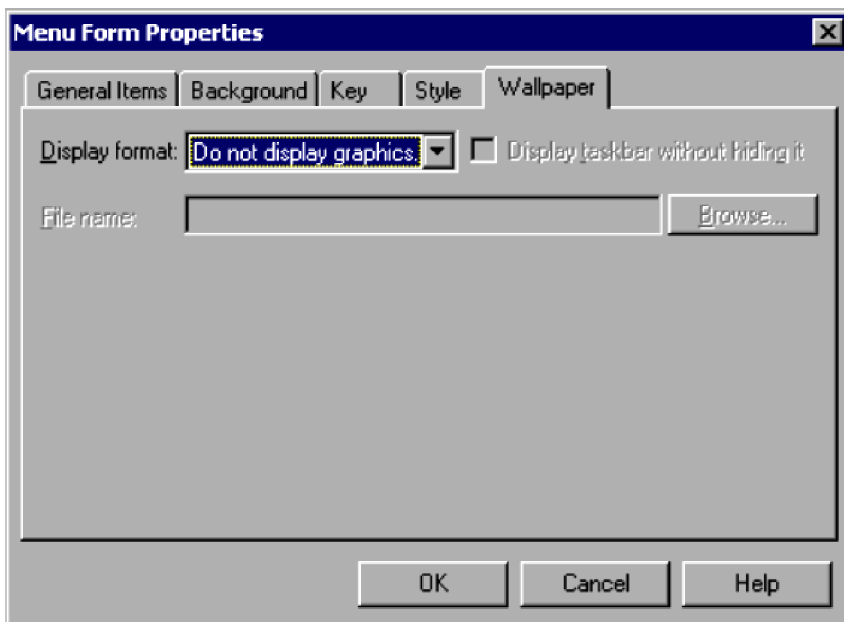
(2) Operations

- Click **OK** to apply the settings and close the Menu Form Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.5 Menu Form Properties (Wallpaper) dialog box

The Menu Form Properties dialog box appears when you select and double-click a menu form in the menu form view of the Script Menu Editor window.

The Menu Form Properties dialog box has five tabs: **General Items**, **Background**, **Key**, **Style**, and **Wallpaper**. Click the **Wallpaper** tab to display the Menu Form Properties (Wallpaper) page.



(1) Components

In the Menu Form Properties (Wallpaper) dialog box, you can set the following items for the wallpaper pasted on the menu form you selected.

Display format

From the drop-down list, select how to display the wallpaper image.

Do not display graphics is set by default.

Display taskbar without hiding it

Displays the image as wallpaper, leaving the task bar visible.

File name

Enter the file name of the image to display as wallpaper.

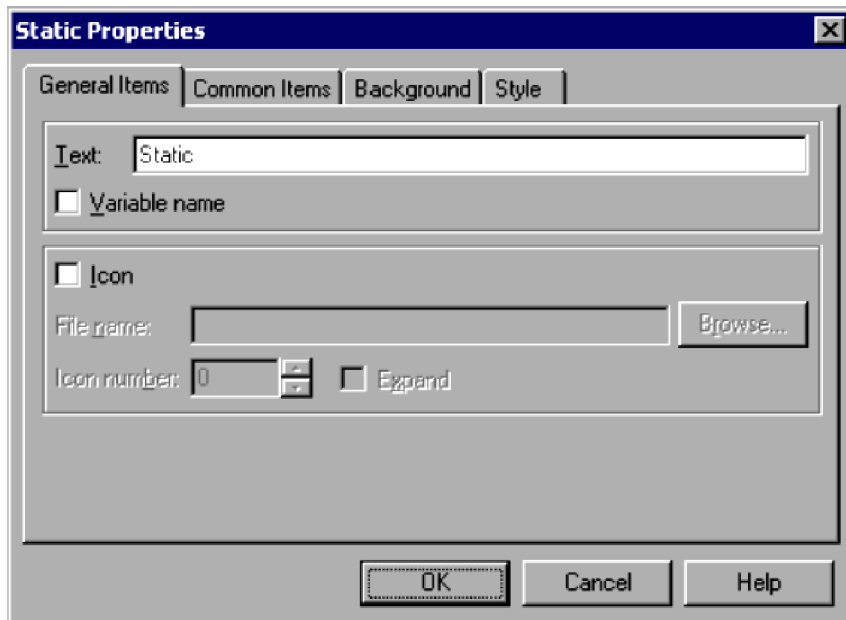
(2) Operations

- Click **OK** to apply the settings and close the Menu Form Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.6 Static Properties (General Items) dialog box

The Static Properties dialog box appears when you select and double-click a static control in the menu form view of the Script Menu Editor window.

The Static Properties dialog box has four tabs: **General Items**, **Common Items**, **Background**, and **Style**. Click the **General Items** tab to display the Static Properties (General Items) page.



(1) Components

In the Static Properties (General Items) dialog box, you can set the following properties for the text displayed on a static control.

Text

Type the text to be displayed. You can specify character strings or a variable name that stores character strings. You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable in **Text**.

Icon

Select this check box to display an icon on a static control.

File name

Enter the icon file name to display on the static control. You can set a file name only if you select **Icon**.

Browse

Click to specify an icon file in the Open File dialog box. This button is available only if you select **Icon**.

Icon number

Specify the icon in the icon file specified in **File name**, using a number beginning from zero. You can set a number only if you select **Icon**.

Expand

Select this check box to display a double-size static control. You can select this check box only if you select **Icon**.

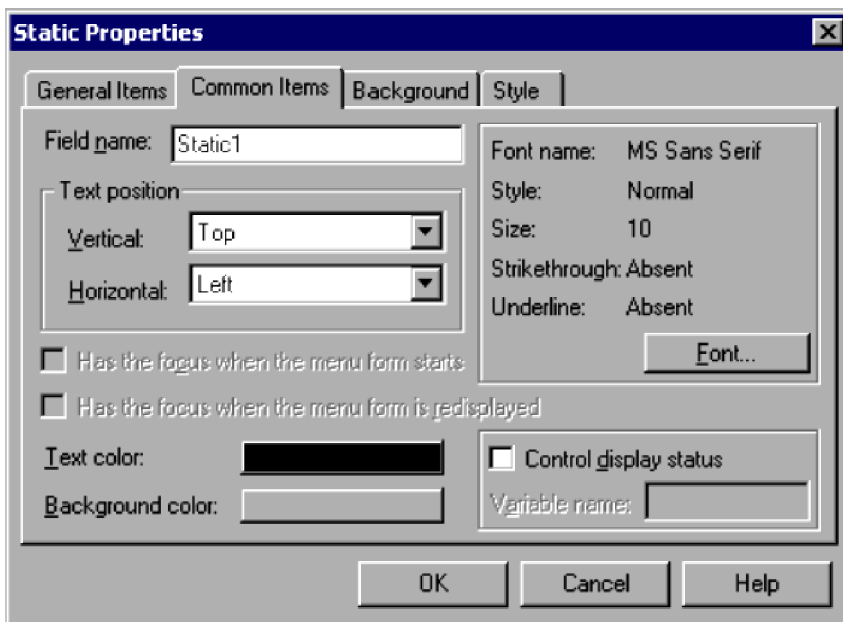
(2) Operations

- Click **OK** to apply the settings and close the Static Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.7 Static Properties (Common Items) dialog box

The Static Properties dialog box appears when you select and double-click a static control in the menu form view of the Script Menu Editor window.

The Static Properties dialog box has four tabs: **General Items**, **Common Items**, **Background**, and **Style**. Click the **Common Items** tab to display the Static Properties (Common Items) page.



(1) Components

In the Static Properties (Common Items) dialog box, you can set the following properties for the text displayed on a static control.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

By default, **Top** is set for **Vertical**, and **Left** is set for **Horizontal**.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the static control. Gray is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button.

Absent is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

Important note

In JP1/Script 06-71 and later versions, **Has the focus when the menu form starts** and **Has the focus when the menu form is redisplayed** are shown in disabled state.

(2) Operations

- Click **OK** to apply the settings and close the Static Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

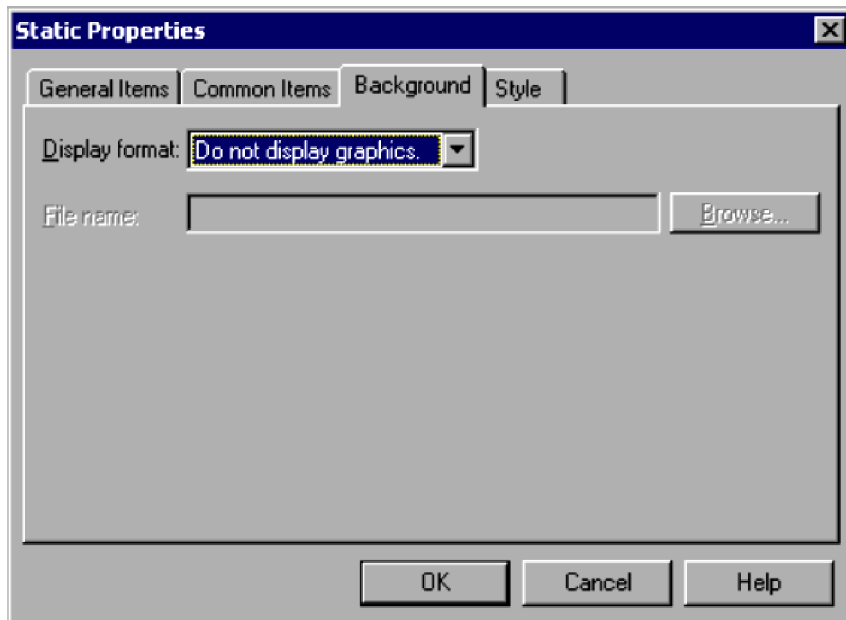
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.8 Static Properties (Background) dialog box

The Static Properties dialog box appears when you select and double-click a static control in the menu form view of the Script Menu Editor window.

The Static Properties dialog box has four tabs: **General Items**, **Common Items**, **Background**, and **Style**. Click the **Background** tab to display the Static Properties (Background) page.



(1) Components

In the Static Properties (Background) dialog box, you can set the following properties for the image displayed in the background of a static control.

Display format

From the drop-down list, select how to display the background image.

Do not display graphics is set by default.

File name

Enter the file name of the image to display in the background.

This field is disabled when **Do not display graphics** is selected in **Display format**.

Browse

Click to select an image file.

This button is disabled when **Do not display graphics** is selected in **Display format**.

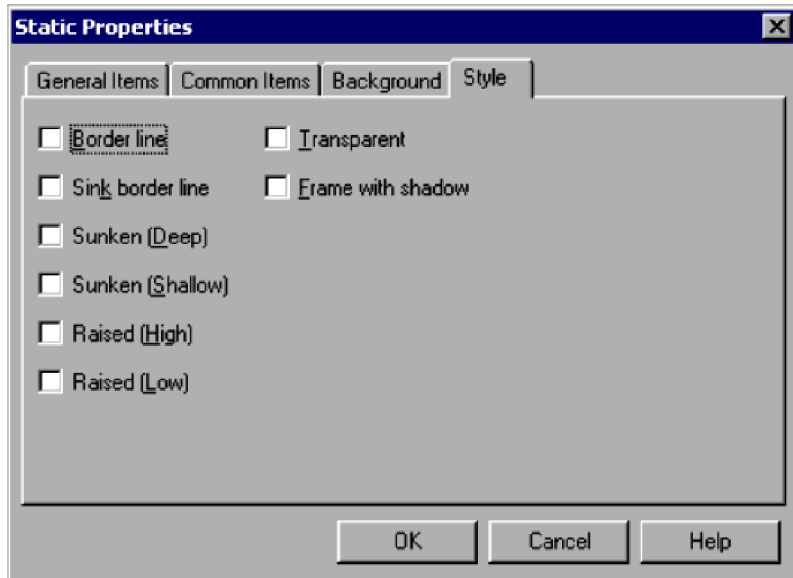
(2) Operations

- Click **OK** to apply the settings and close the Static Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.9 Static Properties (Style) dialog box

The Static Properties dialog box appears when you select and double-click a static control in the menu form view of the Script Menu Editor window.

The Static Properties dialog box has four tabs: **General Items**, **Common Items**, **Background**, and **Style**. Click the **Style** tab to display the Static Properties (Style) page.



(1) Components

In the Static Properties (Style) dialog box, you can set the following style properties for a static control.

Border line

Draws a border around the control.

Sink border line

Draws a grooved border around the control.

Sunken (Deep)

Makes the control appear deeply inset.

Sunken (Shallow)

Makes the control appear slightly inset.

Raised (High)

Makes the control appear prominently raised.

Raised (Low)

Makes the control appear slightly raised.

Transparent

Makes the control transparent.

Frame with shadow

Adds a shadow effect to the control.

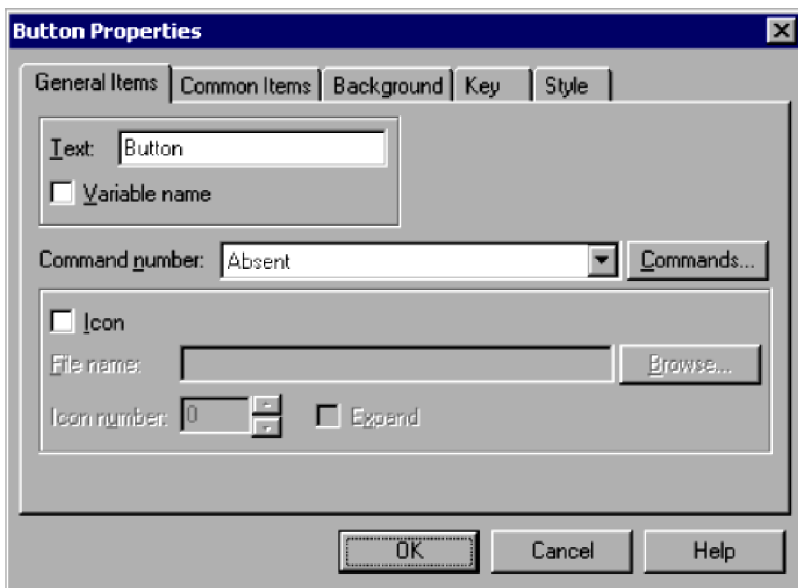
(2) Operations

- Click **OK** to apply the settings and close the Static Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.10 Button Properties (General Items) dialog box

The Button Properties dialog box appears when you select and double-click a button control in the menu form view of the Script Menu Editor window.

The Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **General Items** tab to display the Button Properties (General Items) page.



(1) Components

In the Button Properties (General Items) dialog box, you can set the following items for a button control.

Text

Specify the text to be displayed for the button. You can specify character strings or a variable name that stores character strings.

You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable in **Text**.

Command number

Specify the number of the command to be executed when the user clicks this button.

Command

Click to display the Set Command Properties dialog box and set the command properties.

Icon

Select this check box to display an icon on the button.

File name

Enter the icon file name to display on the button. You can set a file name only if you select **Icon**.

Browse

Click to specify an icon file in the Open File dialog box. This button is available only if you select **Icon**.

Icon number

Specify the icon in the icon file specified in **File name**, using a number beginning from zero. You can set a number only if you select **Icon**.

Expand

Select this check box to display a double-size icon on the button. This check box is available only if you select **Icon**.

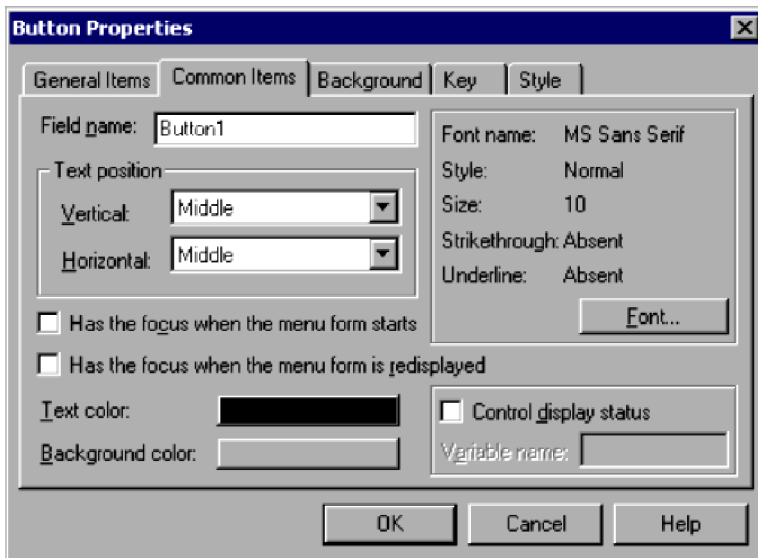
(2) Operations

- Click **OK** to apply the settings and close the Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.11 Button Properties (Common Items) dialog box

The Button Properties dialog box appears when you select and double-click a button control in the menu form view of the Script Menu Editor window.

The Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Common Items** tab to display the Button Properties (Common Items) page.



(1) Components

In the Button Properties (Common Items) dialog box, you can set the following items for a button control.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

By default, **Middle** is set for both **Vertical** and **Horizontal**.

Has the focus when the menu form starts

This button will be given the focus when the menu form is first displayed.

Has the focus when the menu form is redisplayed

This button will be given the focus when the menu form is redrawn.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the button control. Gray is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button.

Absent is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

(2) Operations

- Click **OK** to apply the settings and close the Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

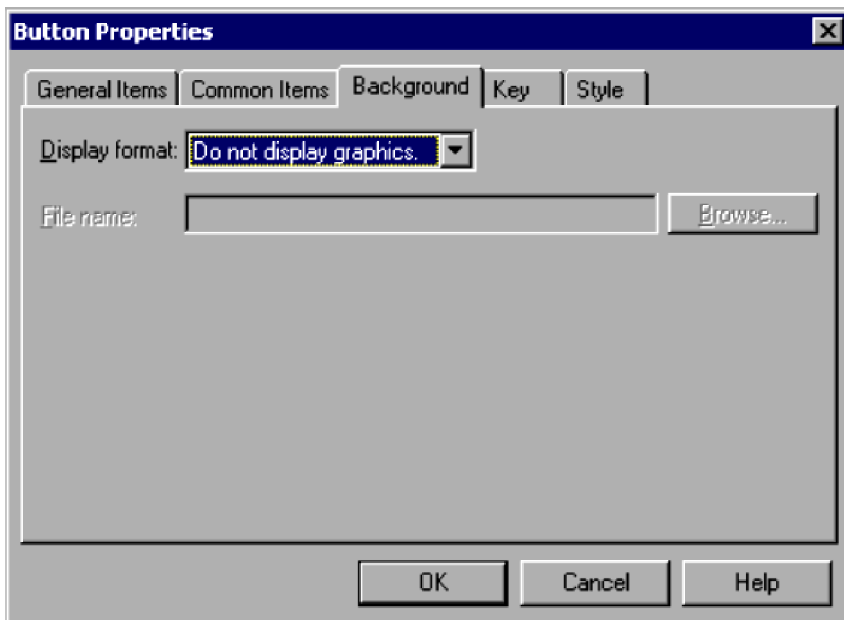
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.12 Button Properties (Background) dialog box

The Button Properties dialog box appears when you select and double-click a button control in the menu form view of the Script Menu Editor window.

The Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Background** tab to display the Button Properties (Background) page.



(1) Components

In the Button Properties (Background) dialog box, you can set the following properties for the background of a button control.

Display format

From the drop-down list, select how to display the background image.

Do not display graphics is set by default.

File name

Enter the file name of the image to display in the background.

This field is disabled when **Do not display graphics** is selected in **Display format**.

Browse

Click to select an image file.

This button is disabled when **Do not display graphics** is selected in **Display format**.

(2) Operations

- Click **OK** to apply the settings and close the Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.13 Button Properties (Key) dialog box

The Button Properties dialog box appears when you select and double-click a button control in the menu form view of the Script Menu Editor window.

The Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Key** tab to display the Button Properties (Key) page.



(1) Components

In the Button Properties (Key) dialog box, you can set the following key properties for a button control.

Specify accelerator keys

Select this check box to specify the accelerator keys.

Ctrl

Sets the **Ctrl** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Shift

Sets the **Shift** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Alt

Sets the **Alt** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Key

Select a key from the drop-down list. You can select a key only if you select **Specify accelerator keys**.

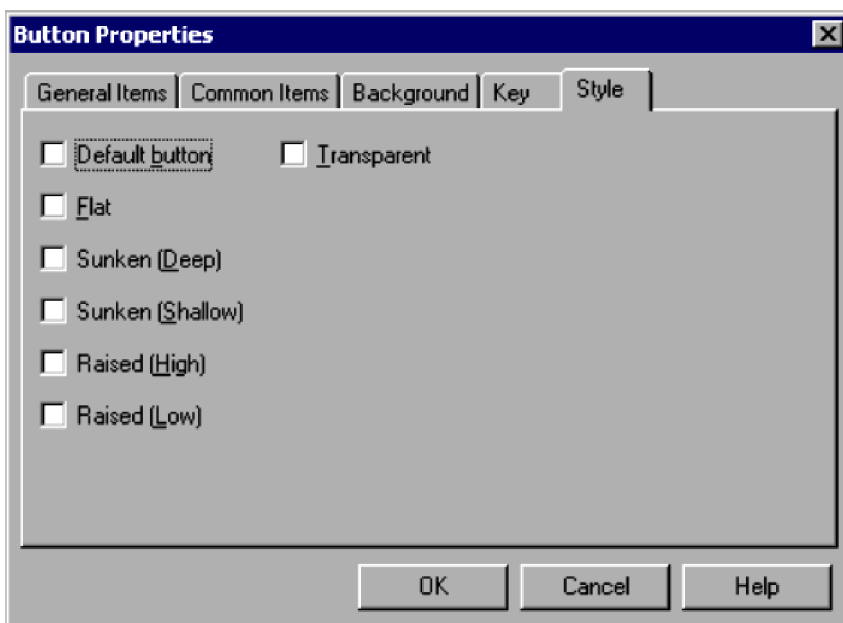
(2) Operations

- Click **OK** to apply the settings and close the Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.14 Button Properties (Style) dialog box

The Button Properties dialog box appears when you select and double-click a button control in the menu form view of the Script Menu Editor window.

The Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Style** tab to display the Button Properties (Style) page.



(1) Components

In the Button Properties (Style) dialog box, you can set the following style properties for a button control.

Default button

Designates the button as the default button. Only one button on a menu form can be the default button.

Flat

Makes the button two-dimensional.

Sunken (Deep)

Makes the button appear deeply inset.

Sunken (Shallow)

Makes the button appear slightly inset.

Raised (High)

Makes the button appear prominently raised.

Raised (Low)

Makes the button appear slightly raised.

Transparent

Makes the button transparent.

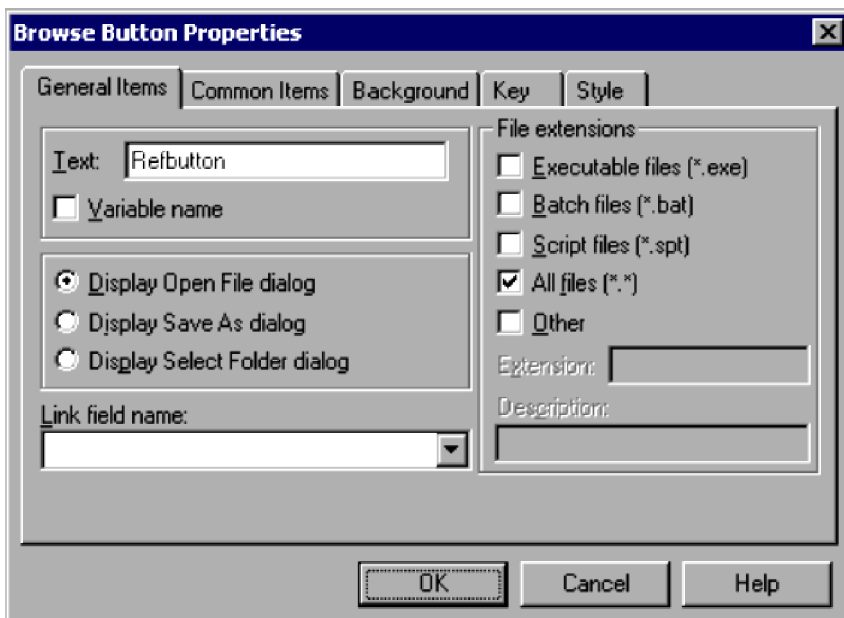
(2) Operations

- Click **OK** to apply the settings and close the Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.15 Browse Button Properties (General Items) dialog box

The Browse Button Properties dialog box appears when you select and double-click a browse button in the menu form view of the Script Menu Editor window.

The Browse Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **General Items** tab to display the Browse Button Properties (General Items) page.



(1) Components

In the Browse Button Properties (General Items) dialog box, you can set the following items for a browse button.

Text

Type the text to be displayed on the browse button. You can specify character strings or a variable name that stores character strings.

You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable in **Text**.

Display Open File dialog

Select this check box to display the Open File dialog box when the user clicks the browse button.

Display Save As dialog

Select this check box to display the Save As dialog box when the user clicks the browse button.

Display Select Folder dialog

Select this check box to display the Select Folder dialog box when the user clicks the browse button.

Link field name

Specify the field name of the control in which the file specified in the Open File dialog box or Save As dialog box or the folder selected in the Select Folder dialog box will be stored.

This item is mandatory.

File extensions

Specify a file format to act as a filter when files are listed in the Open File dialog box or Save As dialog box. Select one of the following:

Executable files <*.exe>

List executable files only.

Batch files <*.bat>

List batch files only.

Script files <*.spt>

List script files only.

All files <*. *>

List all files.

Other

List files other than the above.

Extension

Specify an extension as a filter for listing files in the Open File dialog box or Save As dialog box. This option is available only if you select **Other**.

Description

Write a description of the files to list in the dialog box. This option is available only if you select **Other**.

The **File extensions** area is unavailable if you select the **Display Select Folder dialog** check box.

(2) Operations

- Click **OK** to apply the settings and close the Browse Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

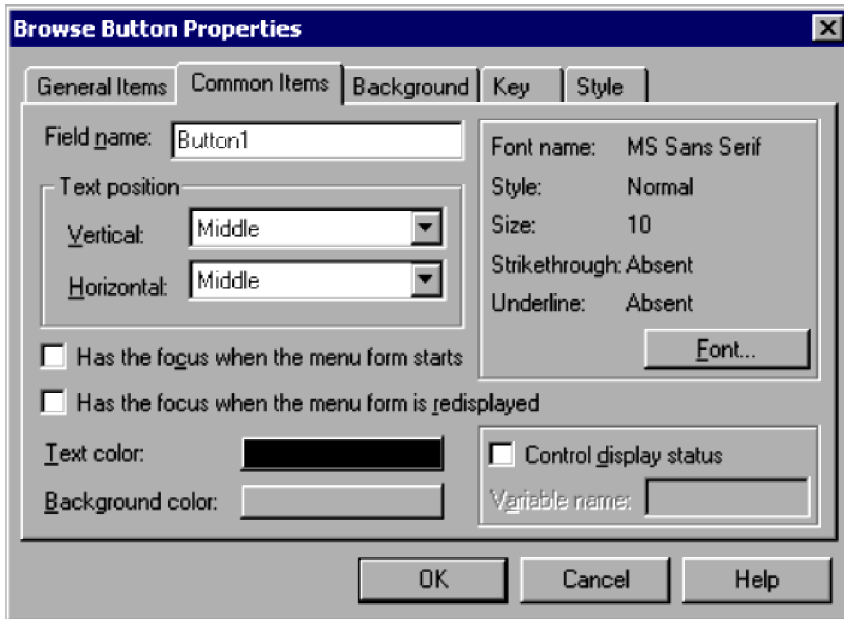
(3) Processing

- When the **Display Open File dialog** or **Display Save As dialog** check box is selected, and the linked field string contains a forward slash (/) or backward slash (\), the selected dialog box does not appear when the user selects the browse button.

4.4.16 Browse Button Properties (Common Items) dialog box

The Browse Button Properties dialog box appears when you select and double-click a browse button in the menu form view of the Script Menu Editor window.

The Browse Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Common Items** tab to display the Browse Button Properties (Common Items) page.



(1) Components

In the Browse Button Properties (Common Items) dialog box, you can set the following properties for the text displayed on a browse button.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

By default, **Middle** is set for both **Vertical** and **Horizontal**.

Has the focus when the menu form starts

The browse button will be given the focus when the menu form is first displayed.

Has the focus when the menu form is redisplayed

The browse button will be given the focus when the menu form is redrawn.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the browse button. Gray is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button.

Absent is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

(2) Operations

- Click **OK** to apply the settings and close the Browse Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

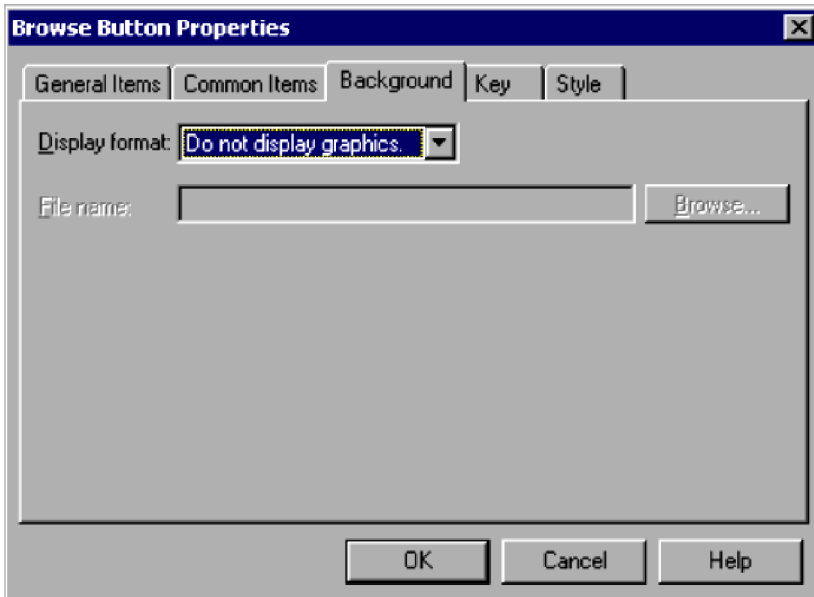
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.17 Browse Button Properties (Background) dialog box

The Browse Button Properties dialog box appears when you select and double-click a browse button in the menu form view of the Script Menu Editor window.

The Browse Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Background** tab to display the Browse Button Properties (Background) page.



(1) Components

In the Browse Button Properties (Background) dialog box, you can set the following properties for the background of a browse button.

Display format

From the drop-down list, select how to display the background image.

Do not display graphics is set by default.

File name

Enter the file name of the image to display in the background.

This field is disabled when **Do not display graphics** is selected in **Display format**.

Browse

Click to select an image file.

This button is disabled when **Do not display graphics** is selected in **Display format**.

(2) Operations

- Click **OK** to apply the settings and close the Browse Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.18 Browse Button Properties (Key) dialog box

The Browse Button Properties dialog box appears when you select and double-click a browse button in the menu form view of the Script Menu Editor window.

The Browse Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Key** tab to display the Browse Button Properties (Key) page.



(1) Components

In the Browse Button Properties (Key) dialog box, you can set the following key properties for a browse button.

Specify accelerator keys

Select this check box to specify the accelerator keys.

Ctrl

Sets the **Ctrl** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Shift

Sets the **Shift** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Alt

Sets the **Alt** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Key

Select a key from the drop-down list. You can select a key only if you select **Specify accelerator keys**.

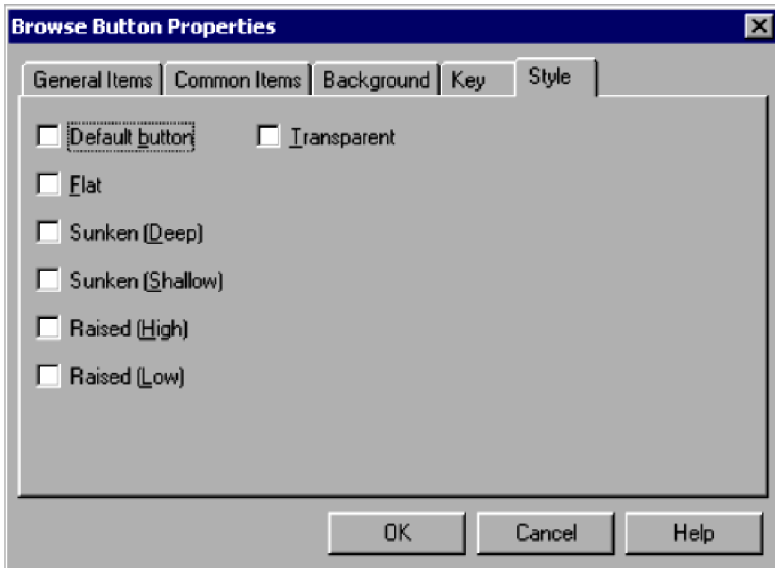
(2) Operations

- Click **OK** to apply the settings and close the Browse Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.19 Browse Button Properties (Style) dialog box

The Browse Button Properties dialog box appears when you select and double-click a browse button in the menu form view of the Script Menu Editor window.

The Browse Button Properties dialog box has five tabs: **General Items**, **Common Items**, **Background**, **Key**, and **Style**. Click the **Style** tab to display the Browse Button Properties (Style) page.



(1) Components

In the Browse Button Properties (Style) dialog box, you can set the following style properties for a browse button.

Default button

Designates the browse button as the default button. Only one button on a menu form can be the default button.

Flat

Makes the browse button two-dimensional.

Sunken (Deep)

Makes the browse button appear deeply inset.

Sunken (Shallow)

Makes the browse button appear slightly inset.

Raised (High)

Makes the browse button appear prominently raised.

Raised (Low)

Makes the browse button appear slightly raised.

Transparent

Makes the browse button transparent.

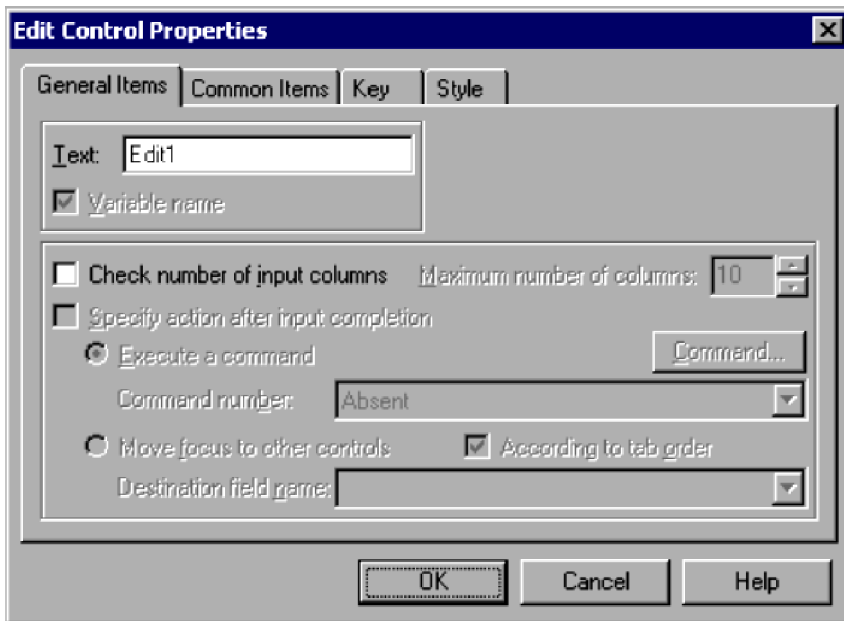
(2) Operations

- Click **OK** to apply the settings and close the Browse Button Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.20 Edit Control Properties (General Items) dialog box

The Edit Control Properties dialog box appears when you select and double-click an edit control in the menu form view of the Script Menu Editor window.

The Edit Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **General Items** tab to display the Edit Control Properties (General Items) page.



(1) Components

In the Edit Control Properties (General Items) dialog box, you can set the following items for an edit control.

Text

Specify the variable that stores the text to be displayed on the edit control.
If you specify an array variable, an error occurs when the menu form is activated.

Check number of input columns

Select this check box to check the number of columns entered by the user.

Maximum number of columns

Specify the maximum number of columns that the user can enter. This field is available only if you select **Check number of input columns**.
Set a number in the range 1 to 1,024. **10** is set by default.

Specify action after input completion

Select this check box to set an action when the user has finished typing in the edit control. This check box is available only if you select **Check number of input columns**.
You cannot select this check box if you selected **Command select field** in the **Style** page.

Execute a command

A command will be executed at completion of input. This button is available only if you select **Specify action after input completion**.

Command number

Specify the number of the command to be executed at completion of input. This field is available only if you select **Execute a command**.

Command

Click to display the Set Command Properties dialog box and set the command properties. This field is available only if you select **Execute a command**.

Move focus to other controls

The focus moves to a different control at completion of input. This button is available only if you select **Specify action after input completion**.

According to tab order

Select this check box to have the focus move in the same sequence as the tab order. This check box is available only if you select **Move focus to other controls**.

Destination field name

Specify the field name of the control to which the focus is to move at completion of input. The drop-down list has the field names of all buttons, browse buttons, edit controls, list controls, and combo boxes on the menu form.

You can set a field name only if you selected **Move focus to other controls** and you did not select **According to tab order**.

(2) Operations

- Click **OK** to apply the settings and close the Edit Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

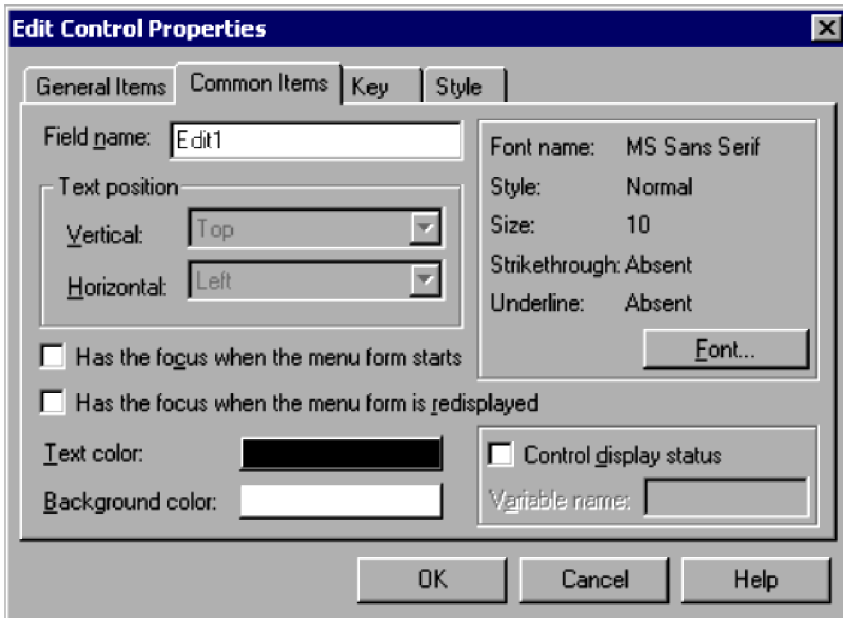
(3) Processing

- The character string entered in the edit control is stored in the variable specified in **Text**.
- Each character entered in an edit control is counted as one column.
- When **According to tab order** is selected, the focus moves according to the tab order after the maximum number of input columns is reached.
- If the field name specified in **Destination field name** does not exist on the menu form, the focus moves according to the tab order.

4.4.21 Edit Control Properties (Common Items) dialog box

The Edit Control Properties dialog box appears when you select and double-click an edit control in the menu form view of the Script Menu Editor window.

The Edit Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Common Items** tab to display the Edit Control Properties (Common Items) page.



(1) Components

In the Edit Control Properties (Common Items) dialog box, you can set the following items for an edit control.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Top is set for **Vertical**, and **Left** is set for **Horizontal**.

Has the focus when the menu form starts

The edit control will be given the focus when the menu form is first displayed.

Has the focus when the menu form is redisplayed

The edit control will be given the focus when the menu form is redrawn.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the edit control. White is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button. **Absent** is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

(2) Operations

- Click **OK** to apply the settings and close the Edit Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

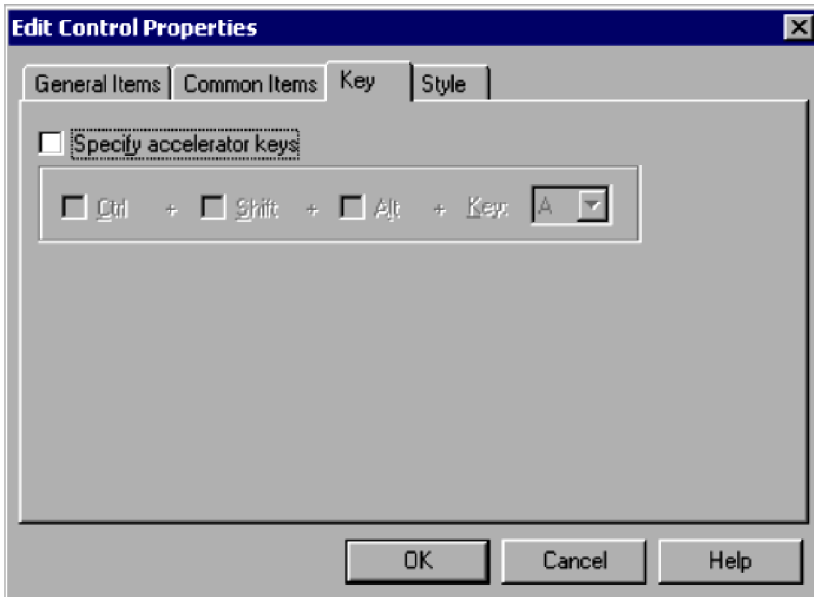
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.22 Edit Control Properties (Key) dialog box

The Edit Control Properties dialog box appears when you select and double-click an edit control in the menu form view of the Script Menu Editor window.

The Edit Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Key** tab to display the Edit Control Properties (Key) page.



(1) Components

In the Edit Control Properties (Key) dialog box, you can set the following key properties for an edit control.

Specify accelerator keys

Select this check box to specify the accelerator keys.

Ctrl

Sets the **Ctrl** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Shift

Sets the **Shift** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Alt

Sets the **Alt** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Key

Select a key from the drop-down list. You can select a key only if you select **Specify accelerator keys**.

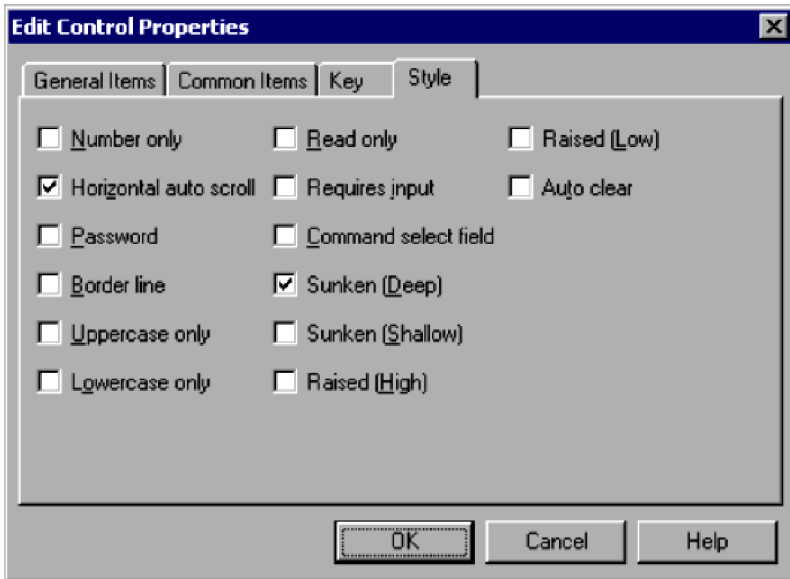
(2) Operations

- Click **OK** to apply the settings and close the Edit Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.23 Edit Control Properties (Style) dialog box

The Edit Control Properties dialog box appears when you select and double-click an edit control in the menu form view of the Script Menu Editor window.

The Edit Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Style** tab to display the Edit Control Properties (Style) page.



(1) Components

In the Edit Control Properties (Style) dialog box, you can set the following items for an edit control.

Number only

Prohibits input of characters other than numerals in the edit control.

Horizontal auto scroll

Automatically scrolls the text if it exceeds the visible area. This check box is selected by default.

Password

Displays all the entered characters as asterisks (*) so that the text cannot be read.

Border line

Draws a border around the edit control.

Uppercase only

Displays all the entered text as uppercase characters.

Lowercase only

Displays all the entered text as lowercase characters.

Read only

Prohibits input and editing of the edit control.

Requires input

Requires the user to type text in the edit control. If you select this item, a warning message appears when no value is entered.

Command select field

Select this check box to use the edit control as the command selection field. When you enter a command number in the command selection field and press the Enter key, the command defined in [4.4.37 Set Command Properties dialog box](#) is executed.

Sunken (Deep)

Makes the edit control appear deeply inset.

Sunken (Shallow)

Makes the edit control appear slightly inset.

Raised (High)

Makes the edit control appear prominently raised.

Raised (Low)

Makes the edit control appear slightly raised.

Auto clear

Clears the edit control each time the menu form is displayed.

(2) Operations

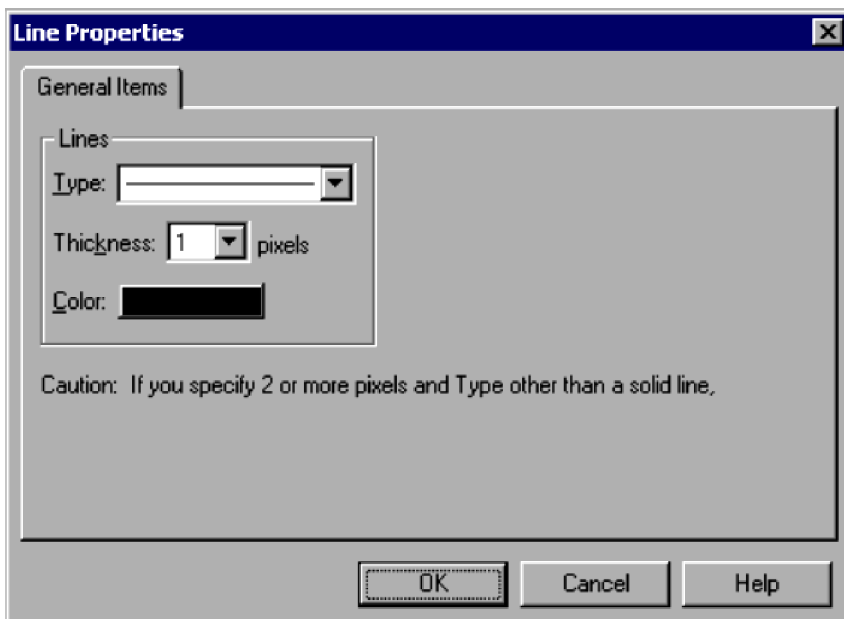
- Click **OK** to apply the settings and close the Edit Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- You cannot select both **Uppercase only** and **Lowercase only**.
- You cannot select both **Read only** and **Auto clear**.

4.4.24 Line Properties (General Items) dialog box

The Line Properties dialog box appears when you select and double-click a line in the menu form view of the Script Menu Editor window.



(1) Components

In the Line Properties (General Items) dialog box, you can set the following items for a line.

Lines

Type

Select the line type from the drop-down list. A solid line is set by default.

Thickness

Specify the line thickness in pixels. **1** is set by default.

Color

Specify the line color. Black is set by default.

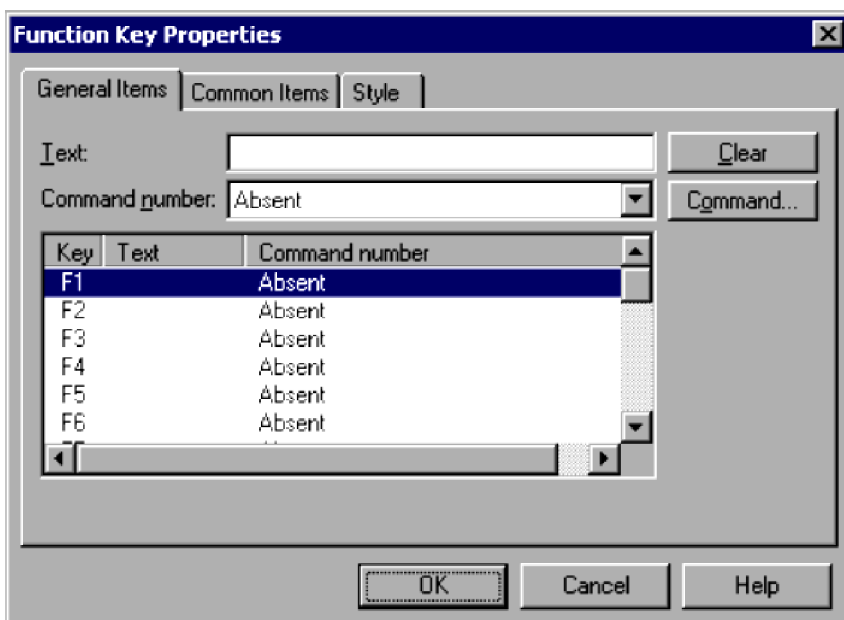
(2) Operations

- Click **OK** to apply the settings and close the Line Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.25 Function Key Properties (General Items) dialog box

The Function Key Properties dialog box appears when you select and double-click a function key control in the menu form view of the Script Menu Editor window.

The Function Key Properties dialog box has three tabs: **General Items**, **Common Items**, and **Style**. Click the **General Items** tab to display the Function Key Properties (General Items) page.



(1) Components

In the Function Key Properties (General Items) dialog box, you can set the following items for a function key control.

Text

Specify the text to be displayed for the function key selected in the list.

Command number

Specify the number of the command to be executed when the user clicks the function key selected in the list.

Clear

Click to clear the data of the function key selected in the list.

Command

Click to set the properties of the command corresponding to the function key selected in the list.

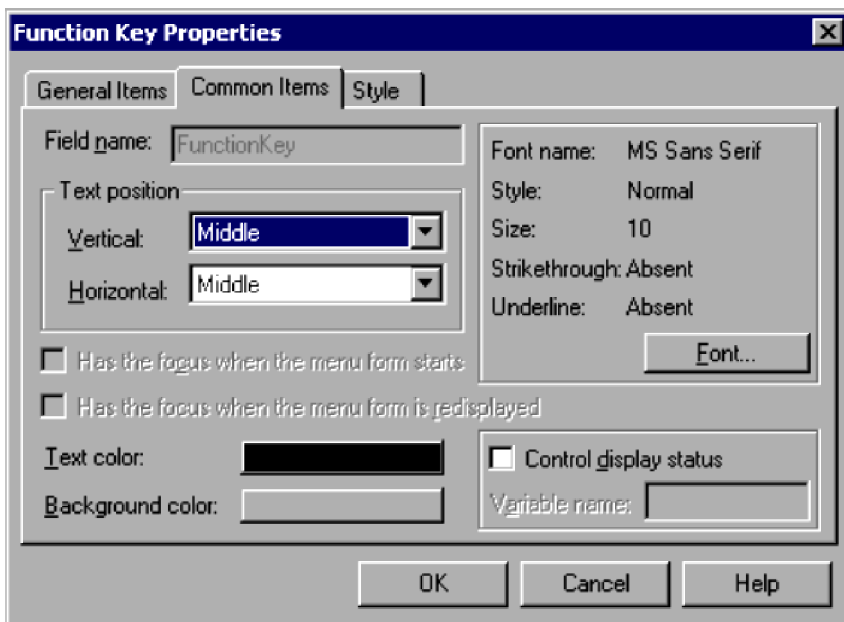
(2) Operations

- Click **OK** to apply the settings and close the Function Key Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.26 Function Key Properties (Common Items) dialog box

The Function Key Properties dialog box appears when you select and double-click a function key control in the menu form view of the Script Menu Editor window.

The Function Key Properties dialog box has three tabs: **General Items**, **Common Items**, and **Style**. Click the **Common Items** tab to display the Function Key Properties (Common Items) page.



(1) Components

In the Function Key Properties (Common Items) dialog box, you can set the following items for a function key control.

Field name

FunctionKey is set.

Text position

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

By default, **Middle** is set for both **Vertical** and **Horizontal**.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the function key. Gray is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button.

Absent is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

Important note

In JP1/Script 06-71 and later versions, **Has the focus when the menu form starts** and **Has the focus when the menu form is redisplayed** are shown in disabled state.

(2) Operations

- Click **OK** to apply the settings and close the Function Key Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

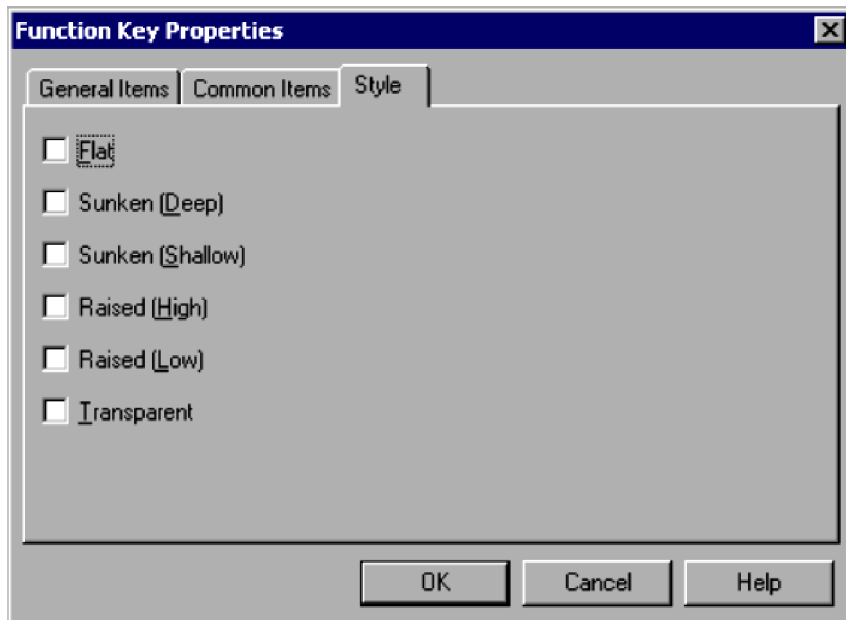
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.27 Function Key Properties (Style) dialog box

The Function Key Properties dialog box appears when you select and double-click a function key control in the menu form view of the Script Menu Editor window.

The Function Key Properties dialog box has three tabs: **General Items**, **Common Items**, and **Style**. Click the **Style** tab to display the Function Key Properties (Style) page.



(1) Components

In the Function Key Properties (Style) dialog box, you can set the following items for a function key control.

Flat

Makes the control two-dimensional.

Sunken (Deep)

Makes the control appear deeply inset.

Sunken (Shallow)

Makes the control appear slightly inset.

Raised (High)

Makes the control appear prominently raised.

Raised (Low)

Makes the control appear slightly raised.

Transparent

Makes the control transparent.

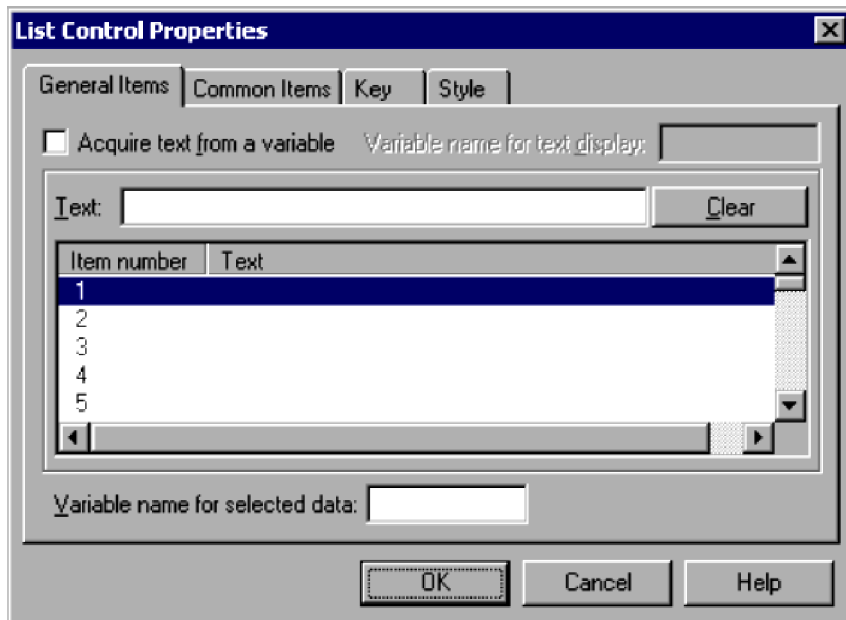
(2) Operations

- Click **OK** to apply the settings and close the Function Key Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.28 List Control Properties (General Items) dialog box

The List Control Properties dialog box appears when you select and double-click a list control in the menu form view of the Script Menu Editor window.

The List Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **General Items** tab to display the List Control Properties (General Items) page.



(1) Components

In the List Control Properties (General Items) dialog box, you can set the following items for a list control.

Acquire text from a variable

Select this check box to acquire the text that is to appear in the list from a variable.

Variable name for text display

Specify a variable name for setting the text in the list.

You can enter a variable only if you select **Acquire text from a variable**. If you specify a two-dimensional array variable, an error occurs when the menu form is activated.

Text

Type the text that is to appear in the list. You cannot set **Text** if you select **Acquire text from a variable**.

Clear

Deletes the data from the list. You cannot choose **Clear** if you select **Acquire text from a variable**.

Variable name for selected data

Specify a variable name for setting the list status.

In the following cases, an error occurs when the menu form is activated:

- **Acquire text from a variable** is unselected and the variable name is an array variable.
- **Acquire text from a variable** is selected, the user can select only one item in the list, and the variable name is an array variable.
- **Acquire text from a variable** is selected, the user can select multiple items in the list, and the variable name is a two-dimensional array variable.

(2) Operations

- Click **OK** to apply the settings and close the List Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- When **Acquire text from a variable** is selected, any elements without a value in the variable specified in **Variable name for text display** do not appear in the list.
- The following values are set in the variable specified in **Variable name for selected data**:

When **Acquire text from a variable** is selected:

Index numbers of the selected data in the variable specified in **Variable name for text display**.

When **Acquire text from a variable** is unselected:

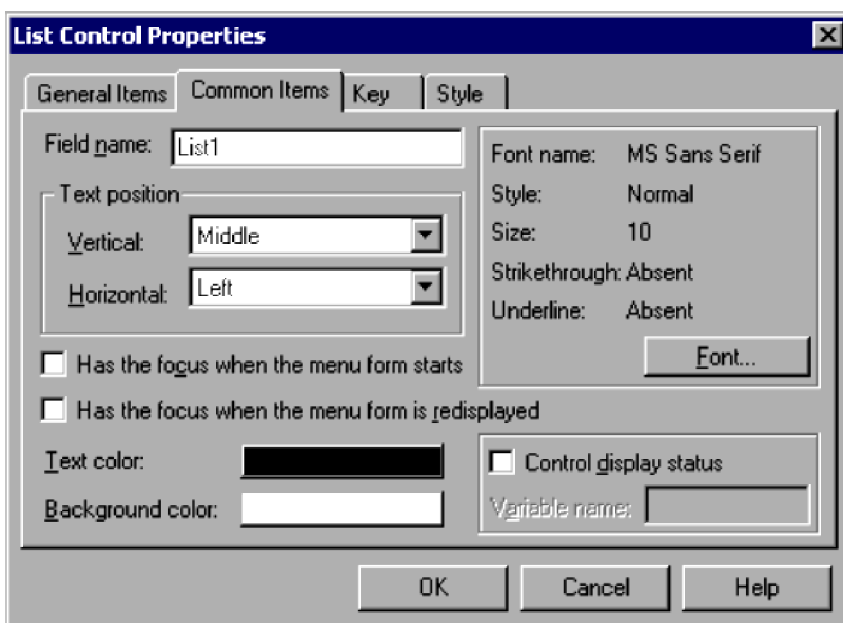
Item numbers if only a single item can be selected in the list.

A numeric string in the form 001002003 ... 100 if the list allows multiple items to be selected.

4.4.29 List Control Properties (Common Items) dialog box

The List Control Properties dialog box appears when you select and double-click a list control in the menu form view of the Script Menu Editor window.

The List Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Common Items** tab to display the List Control Properties (Common Items) page.



(1) Components

In the List Control Properties (Common Items) dialog box, you can set the following items for a list control.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

By default, **Middle** is set for **Vertical**, and **Left** is set for **Horizontal**.

Has the focus when the menu form starts

The list control will be given the focus when the menu form is first displayed.

Has the focus when the menu form is redisplayed

The list control will be given the focus when the menu form is redrawn.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the list control. White is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button.

Absent is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

(2) Operations

- Click **OK** to apply the settings and close the List Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

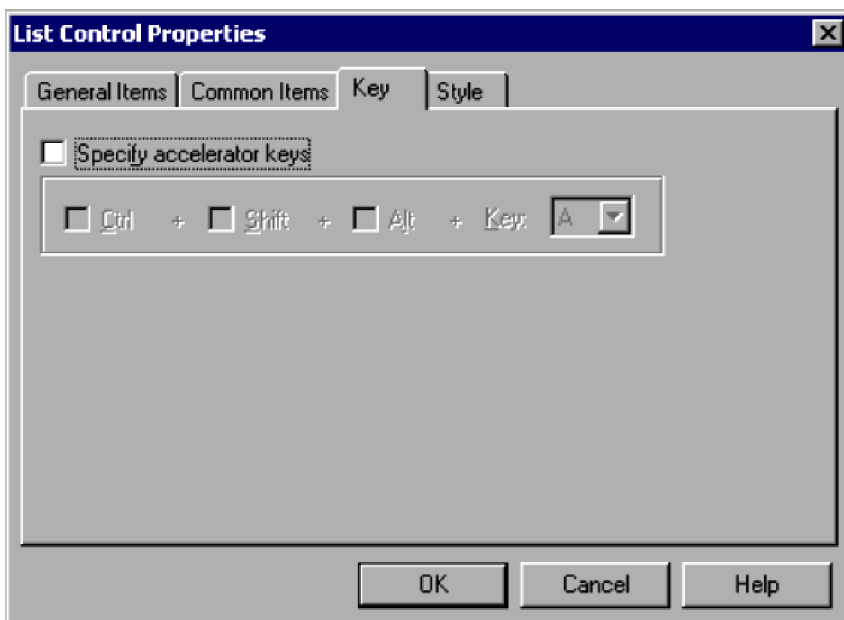
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.30 List Control Properties (Key) dialog box

The List Control Properties dialog box appears when you select and double-click a list control in the menu form view of the Script Menu Editor window.

The List Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Key** tab to display the List Control Properties (Key) page.



(1) Components

In the List Control Properties (Key) dialog box, you can set the following key properties for a list control.

Specify accelerator keys

Select this check box to specify the accelerator keys.

Ctrl

Sets the **Ctrl** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Shift

Sets the **Shift** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Alt

Sets the **Alt** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Key

Select a key from the drop-down list. You can select a key only if you select **Specify accelerator keys**.

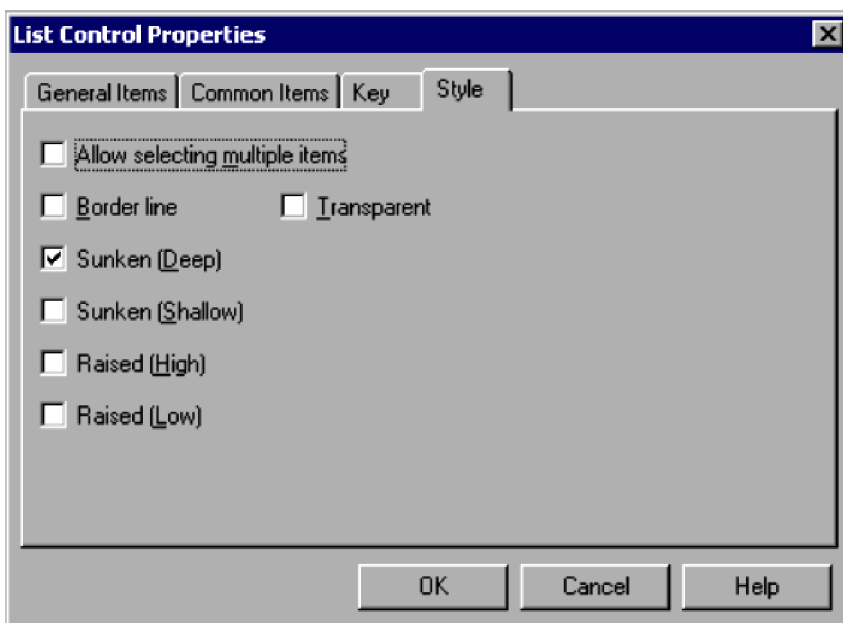
(2) Operations

- Click **OK** to apply the settings and close the List Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.31 List Control Properties (Style) dialog box

The List Control Properties dialog box appears when you select and double-click a list control in the menu form view of the Script Menu Editor window.

The List Control Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Style** tab to display the List Control Properties (Style) page.



(1) Components

In the List Control Properties (Style) dialog box, you can set the following items for a list control.

Allow selecting multiple items

Allows the user to select multiple items.

Border line

Draws a border around the list control.

Sunken (Deep)

Makes the list control appear deeply inset. This check box is selected by default.

Sunken (Shallow)

Makes the list control appear slightly inset.

Raised (High)

Makes the list control appear prominently raised.

Raised (Low)

Makes the list control appear slightly raised.

Transparent

Makes the list control transparent.

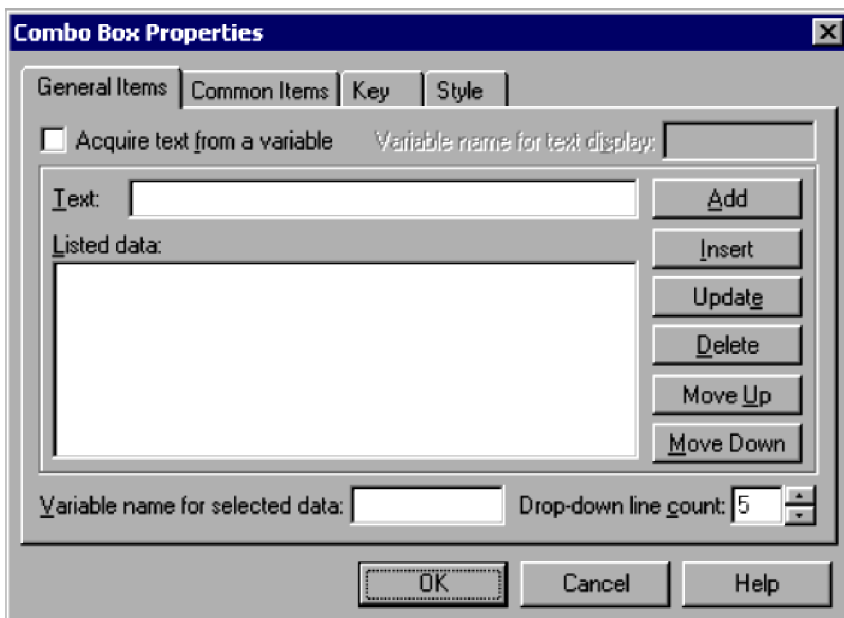
(2) Operations

- Click **OK** to apply the settings and close the List Control Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.32 Combo Box Properties (General Items) dialog box

The Combo Box Properties dialog box appears when you select and double-click a combo box in the menu form view of the Script Menu Editor window.

The Combo Box Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **General Items** tab to display the Combo Box Properties (General Items) page.



(1) Components

In the Combo Box Properties (General Items) dialog box, you can set the following items for a combo box.

Acquire text from a variable

Select this check box to acquire the text to appear in the combo box from a variable.

Variable name for text display

Specify a variable name for setting the text to appear in the combo box.

You can enter a variable only if you select **Acquire text from a variable**. If you specify a two-dimensional array variable, an error occurs when the menu form is activated.

Text

Type the text to appear in the combo box. You cannot set **Text** if you select **Acquire text from a variable**.

Listed data

Lists the data to appear in the list attached to the combo box. The list is disabled if you select **Acquire text from a variable**.

Add

Adds the data specified in **Text** to the end of the list displayed in **Listed data**.

This button is disabled if you select **Acquire text from a variable**.

Insert

Inserts the data specified in **Text** above the data selected in **Listed data**.

This button is disabled if you select **Acquire text from a variable**.

Update

Updates the data selected in **Listed data** with the data specified in **Text**.

This button is disabled if you select **Acquire text from a variable**.

Delete

Deletes data selected in the list. This button is disabled if you select **Acquire text from a variable**.

Move Up

Moves data selected in the list up one line. This button is disabled if you select **Acquire text from a variable**.

Move Down

Moves data selected in the list down one line. This button is disabled if you select **Acquire text from a variable**.

Variable name for selected data

Specify a variable name for setting the status of the combo box. If you specify an array variable, an error occurs when the menu form is activated.

Drop-down line count

Specify the number of lines to appear in the drop-down list attached to the combo box.

Set a number in the range 1 to 100. **5** is set by default.

(2) Operations

- Click **OK** to apply the settings and close the Combo Box Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

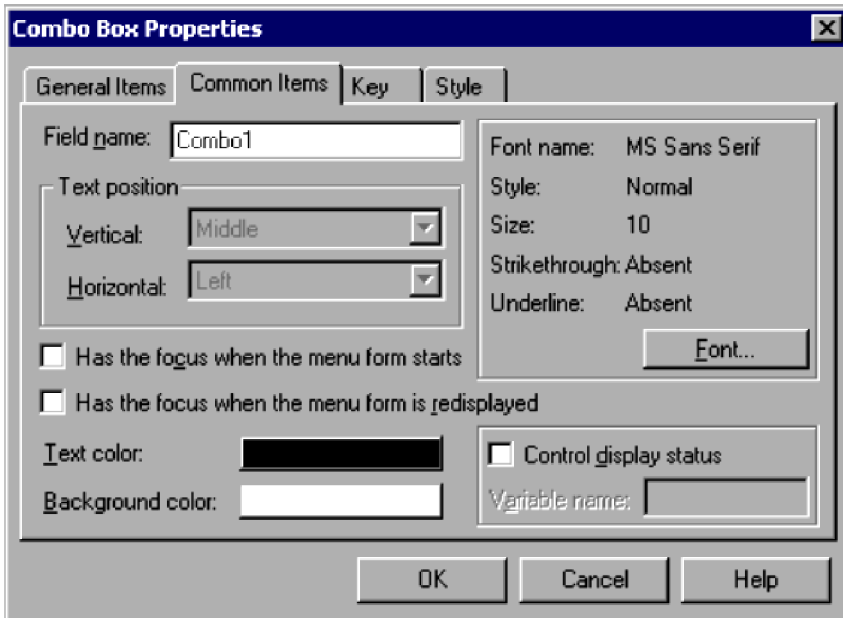
(3) Processing

- When **Acquire text from a variable** is selected, any elements without a value in the variable specified in **Variable name for text display** do not appear in the drop-down list.
- The variable specified in **Variable name for selected data** stores the text that the user selects or enters in the combo box.

4.4.33 Combo Box Properties (Common Items) dialog box

The Combo Box Properties dialog box appears when you select and double-click a combo box in the menu form view of the Script Menu Editor window.

The Combo Box Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Common Items** tab to display the Combo Box Properties (Common Items) page.



(1) Components

In the Combo Box Properties (Common Items) dialog box, you can set the following items for a combo box.

Field name

Type the name of the field, complying with the variable naming conventions.

Text position

Middle is set for **Vertical**, and **Left** is set for **Horizontal**.

Has the focus when the menu form starts

The combo box will be given the focus when the menu form is first displayed.

Has the focus when the menu form is redisplayed

The combo box will be given the focus when the menu form is redrawn.

Text color

Set the text color. Black is set by default.

Background color

Set the background color of the combo box. White is set by default.

Font name

Shows the font currently selected. To change the font, click the **Font** button.

MS Sans Serif is set by default.

Style

Shows the style of the selected font. To change the style, click the **Font** button.

Normal is set by default.

Size

Shows the size of the selected font. To change the size, click the **Font** button.

10 is set by default.

Strikethrough

Shows whether to use strikethrough characters. To change the strikethrough setting, click the **Font** button.

Absent is set by default.

Underline

Shows whether to underline the text. To change the underline setting, click the **Font** button. **Absent** is set by default.

Control display status

Choose whether to control the display status of the control.

Variable name

Specify a variable name for storing the values that indicate the control's display status.

You can set this field only if you select the **Control display status** check box. If you specify an array variable, an error occurs when the menu form is activated.

The following values indicate the display status:

DISABLE

Disable the control on the menu form.

HIDE

Do not show the control on the menu form.

Other value

Enable the control on the menu form.

(2) Operations

- Click **OK** to apply the settings and close the Combo Box Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

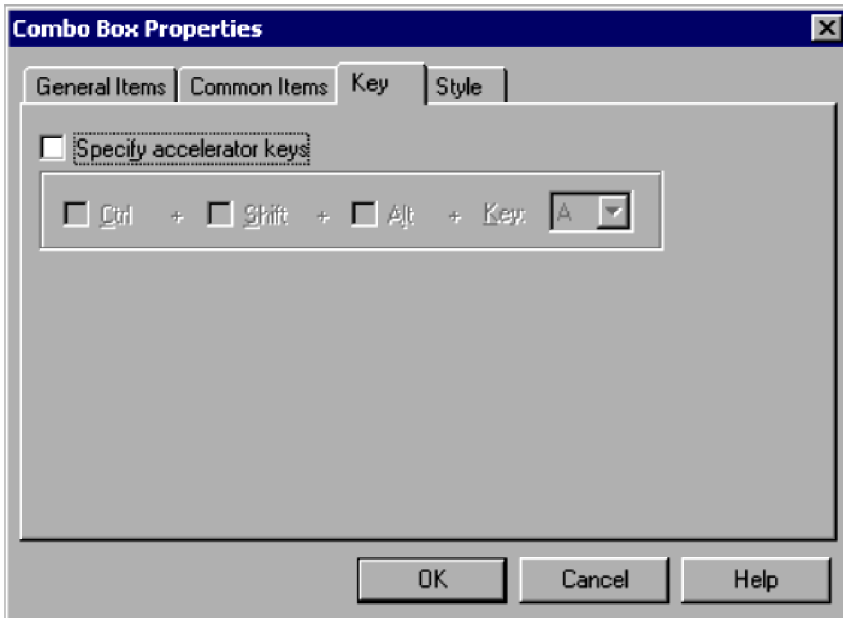
(3) Processing

- When the **Control display status** check box is unselected, the control is shown as enabled on the menu form.
- The **Control display status** setting is invalid for text display.

4.4.34 Combo Box Properties (Key) dialog box

The Combo Box Properties dialog box appears when you select and double-click a combo box in the menu form view of the Script Menu Editor window.

The Combo Box Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Key** tab to display the Combo Box Properties (Key) page.



(1) Components

In the Combo Box Properties (Key) dialog box, you can set the following items for a combo box.

Specify accelerator keys

Select this check box to specify the accelerator keys.

Ctrl

Sets the **Ctrl** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Shift

Sets the **Shift** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Alt

Sets the **Alt** key as a modifier key. This option is available only if you select **Specify accelerator keys**.

Key

Select a key from the drop-down list. You can select a key only if you select **Specify accelerator keys**.

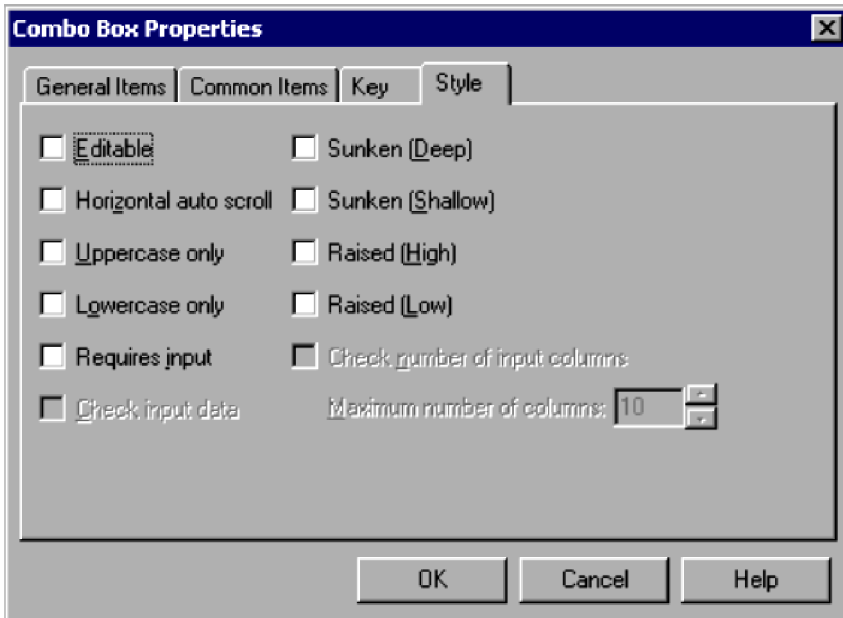
(2) Operations

- Click **OK** to apply the settings and close the Combo Box Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.35 Combo Box Properties (Style) dialog box

The Combo Box Properties dialog box appears when you select and double-click a combo box in the menu form view of the Script Menu Editor window.

The Combo Box Properties dialog box has four tabs: **General Items**, **Common Items**, **Key**, and **Style**. Click the **Style** tab to display the Combo Box Properties (Style) page.



(1) Components

In the Combo Box Properties (Style) dialog box, you can set the following items for a combo box.

Editable

Creates a combo box that can be edited.

Horizontal auto scroll

Automatically scrolls the text if it exceeds the visible area.

Uppercase only

Displays all the entered text as uppercase characters.

Lowercase only

Displays all the entered text as lowercase characters.

Requires input

Requires the user to type text in the combo box.

Check input data

Returns an error if the user enters data that is not in the list attached to the combo box. This check box is available only if you select the **Editable** check box.

Sunken (Deep)

Makes the combo box appear deeply inset.

Sunken (Shallow)

Makes the combo box appear slightly inset.

Raised (High)

Makes the combo box appear prominently raised.

Raised (Low)

Makes the combo box appear slightly raised.

Check number of input columns

Select this check box to check the number of columns entered by the user.

This check box is available only if you select the **Editable** check box.

Maximum number of columns

Specify the maximum number of columns that the user can enter. This field is available only if you select **Check number of input columns**.

Set a number in the range 1 to 1,024. **10** is set by default.

(2) Operations

- Click **OK** to apply the settings and close the Combo Box Properties dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

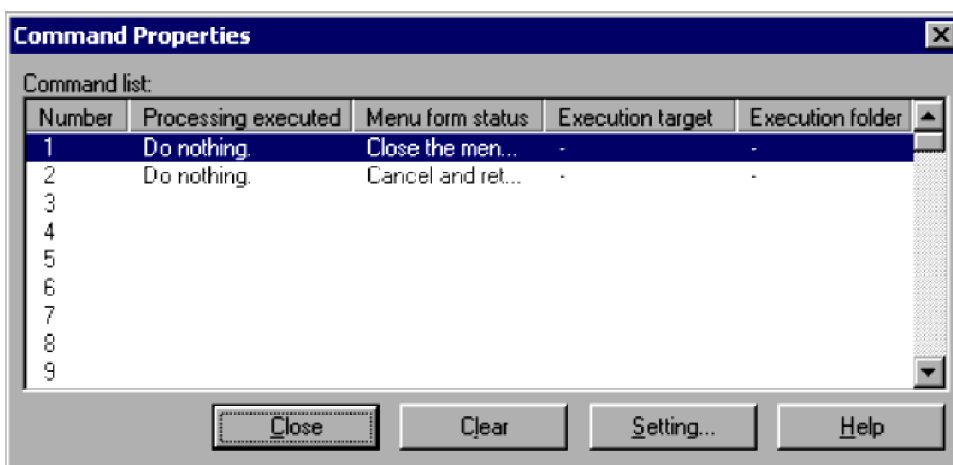
- You cannot select both **Uppercase only** and **Lowercase only**.
- The data check specified in **Check input data** is case insensitive.

4.4.36 Command Properties dialog box

The Command Properties dialog box defines the command execution when a user clicks a button control.

You can choose whether to show or hide this dialog box by choosing **View, Command Properties** in the Script Menu Editor window.

By default, the first command in the list has **Do nothing** in the **Processing executed** field, and **Close the menu form** in the **Menu form status** field. The second command has **Do nothing** and **Cancel and return to the previous menu form**, respectively.



(1) Components

In the Command Properties dialog box, you can set the following items regarding the execution of a command.

Command list

Lists the commands that you set in the Set Command Properties dialog box. The list contains the following items:

Number

Shows the command number.

Processing executed

Shows the command action.

Menu form status

Shows the status of the menu form after the command is executed.

Execution target

Shows the entity on which the command is executed.

Execution folder

Shows the command execution folder.

Close

Closes the Command Properties dialog box.

Clear

Deletes the properties defined for a command selected in the list.

Setting

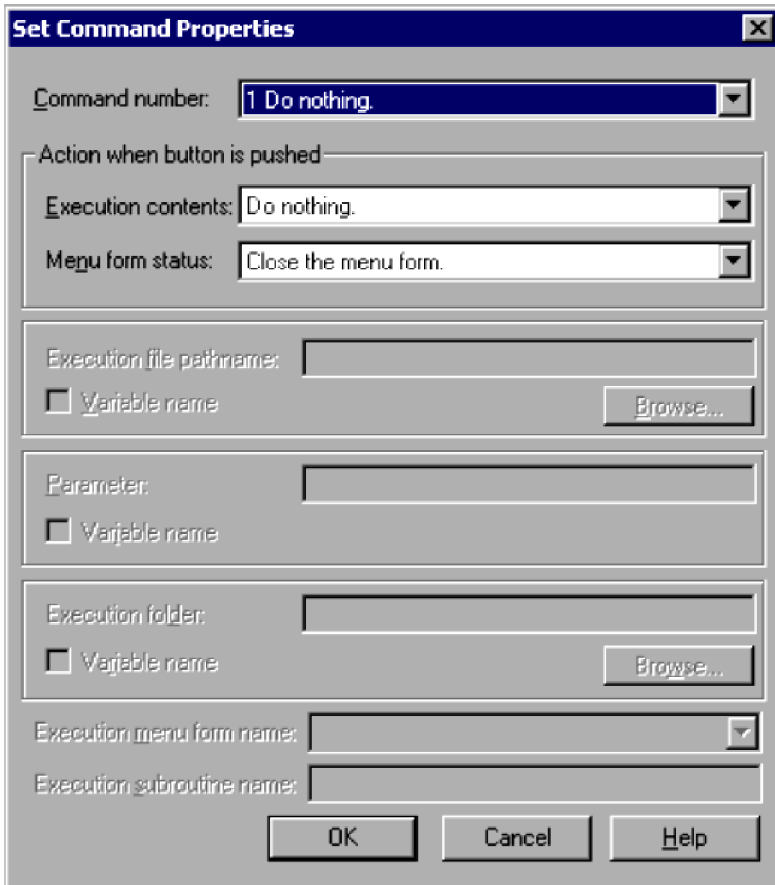
Click to define the command properties. The Set Command Properties dialog box appears.

(2) Operations

- Click **Close** to close the Command Properties dialog box.

4.4.37 Set Command Properties dialog box

The Set Command Properties dialog box appears when you choose **Tools, Set Command Properties**. In this dialog box, you can define the command properties that apply when a user clicks a button control or function key control, or enters data in the edit control.



(1) Components

In the Set Command Properties dialog box, you can set the following items for a command.

Command number

Specify the number of the command.

Action when button is pushed

Execution contents and Menu form status

From the drop-down list, select the action at command execution and the status after the command is executed.

The default settings are **Execute an external command** and **Close the menu form**, respectively.

See (3) below for further details on defining the command action.

Execution file pathname

Specify the file path for execution. You can specify character strings or a variable name that stores character strings. If the character string you specified or the character string stored in the variable is an invalid file path, an error occurs.

You can specify a file path only if you set **Execute an external command** in **Execution contents**.

You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable name in **Execution file pathname**.

Browse

Click to specify a file path in the Open File dialog box.

Parameter

Specify a parameter to pass to the executable command. You can specify character strings or a variable name that stores character strings.

If you specify any of the character strings listed below as a parameter, the setting is treated as the default window display setting for the application to be started. This parameter is not applicable to applications that display windows explicitly.

You cannot specify an array variable for a variable name.

Parameter	Meaning
"/SPT:HIDE"	Hides the application window.
"/SPT:MIN"	Minimizes the application window to an icon.
"/SPT:MAX"	Maximizes the application window.

You can specify a parameter only if you set **Execute an external command** in **Processing executed**.

Variable name

Select this check box if you want to specify a variable name in **Parameter**.

Note that you can specify only one variable. To pass multiple parameters as variables, you must set the multiple values in a single variable by using the space as a delimiter between the values.

Execution folder

Specify the execution folder for the command. You can specify character strings or a variable name that stores character strings.

You can specify an execution folder only if you set **Execute an external command** in **Processing executed**. If you do not specify a folder, the current folder is assumed internally.

You cannot specify an array variable for a variable name.

Variable name

Select this check box if you want to specify a variable name in **Execution folder**.

Browse

Click to specify an execution folder in the Select Folder dialog box.

Execution menu form name

Specify the name of the menu form to be activated.

You can specify a menu form name only if you set **Display a menu form in the file** in **Execution contents**.

Execution subroutine name

Specify the name of the subroutine to be executed.

You can specify a subroutine name only if you set **Call a subroutine in the file** in **Execution contents**.

(2) Operations

- Click **OK** to apply the settings and close the Set Command Properties dialog box.
- Click **Cancel** to close the dialog box without changing the command properties.

(3) Button actions

The following tables show the possible combinations of **Execution contents** and **Menu form status** settings in the **Action when button is pushed** field, and the system processing for each combination.

(a) Execution contents = Execute an external command

Menu form status	Specifiable	System processing
Close the menu form.	Yes	Executes the external command and closes the menu form without waiting for the command to complete.
Cancel and return to the previous menu form.	No	--
Go back to the previous menu form.	Yes	Executes the external command and returns to the previous menu form without waiting for the command to complete.
Wait for termination with the information displayed.	Yes	Executes the external command and leaves the menu form in disabled state until the action completes.
Wait for termination with the information hidden.	Yes	Executes the external command and hides the menu form until the action completes.
Wait with the window minimized.	Yes	Executes the external command and minimizes the menu form until the action completes.
Place it in idle status.	Yes	Simply executes an external command but does not wait for completion.

Legend:

Yes: Combination that can be specified.

No: Combination that cannot be specified.

--: Not applicable

(b) Execution contents = Display a menu form in the file

Menu form status	Specifiable	System processing
Close the menu form.	No	--
Cancel and return to the previous menu form.	No	--
Go back to the previous menu form.	No	--
Wait for termination with the information displayed.	Yes	Displays the specified menu form and leaves the active menu form in disabled state until the action completes.
Wait for termination with the information hidden.	Yes	Displays the specified menu form and hides the active menu form until the action completes.
Wait with the window minimized.	No	--
Place it in idle status.	No	--

Legend:

Yes: Combination that can be specified.

No: Combination that cannot be specified.

--: Not applicable

(c) Execution contents = Call a subroutine in the file

Menu form status	Specifiable	System processing
Close the menu form.	Yes	Calls the specified subroutine and leaves the menu form in disabled state until the action completes. If the subroutine is a <code>Function</code> statement, ends the menu form if the return value is <code>True</code> , otherwise, does nothing.
Cancel and return to the previous menu form.	No	--

Menu form status	Specifiable	System processing
Go back to the previous menu form.	Yes	Calls the specified subroutine and leaves the menu form in disabled state until the action completes. If the subroutine is a <code>Function</code> statement, returns to the previous menu form if the return value is <code>True</code> , otherwise, does nothing.
Wait for termination with the information displayed.	Yes	Calls the specified subroutine and leaves the menu form in disabled state until the action completes.
Wait for termination with the information hidden.	Yes	Calls the specified subroutine and hides the menu form until the action completes.
Wait with the window minimized.	No	--
Place it in idle status.	No	--

Legend:

Yes: Combination that can be specified.

No: Combination that cannot be specified.

--: Not applicable

(d) Execution contents = Switch function keys

Menu form status	Specifiable	System processing
Close the menu form.	No	--
Cancel and return to the previous menu form.	No	--
Go back to the previous menu form.	No	--
Wait for termination with the information displayed.	No	--
Wait for termination with the information hidden.	No	--
Wait with the window minimized.	No	--
Place it in idle status.	No	--

Legend:

Yes: Combination that can be specified.

No: Combination that cannot be specified.

--: Not applicable

(e) Execution contents = Do nothing

Menu form status	Specifiable	System processing
Close the menu form.	Yes	Closes the menu form.
Cancel and return to the previous menu form.	Yes	Returns to the previous menu form without checking the entered data or assigning values to variables.
Go back to the previous menu form.	Yes	Returns to the previous menu form.
Wait for termination with the information displayed.	No	--
Wait for termination with the information hidden.	No	--
Wait with the window minimized.	Yes	Minimizes the menu form.
Place it in idle status.	Yes	Does nothing.

Legend:

Yes: Combination that can be specified.

No: Combination that cannot be specified.

--: Not applicable

(4) Processing

- When you set **Execute an external command** in **Execution contents**, the following values are stored as reserved variables.

Menu form status	Reserved variable	Stored value
Place it in idle status.	<code>_EXEC_RTN_</code>	None
	<code>_EXEC_ID_</code>	Identifier of the executable file
All other values	<code>_EXEC_RTN_</code>	Exit code of external command

- A message box appears if the external command executed from the **Menu** command returns a non-zero exit code. If you do not want to display a message box, set 0 as the value of the registry shown below.

Registry key

HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\SPTX

Value name

Menu_EmgMsgBox

Value data type

DWORD

Values

0: Do not display a message box.

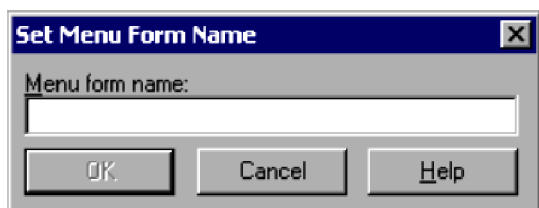
1: Display a message box.

When the setting takes effect

The setting takes effect the next time the script file is executed.

4.4.38 Set Menu Form Name dialog box

The Set Menu Form Name dialog box appears when you choose **Edit, Create Menu Form**. This dialog box is for naming a new menu form.



(1) Components

In the Set Menu Form Name dialog box, you can set the following items.

Menu form name

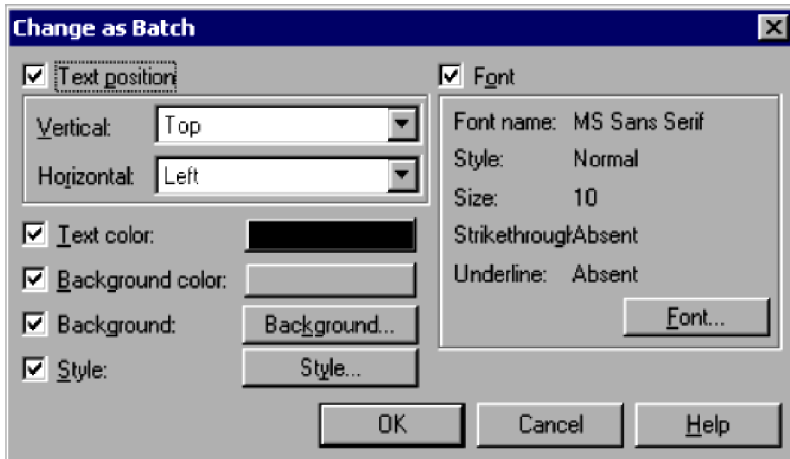
Type the name of the menu form you are creating.

(2) Operations

- Click **OK** to apply the setting and close the Set Menu Form Name dialog box.
- Click **Cancel** to close the dialog box without naming the menu form.

4.4.39 Change as Batch dialog box

The Change as Batch dialog box appears when you choose **Edit, Change as Batch**. In this dialog box, you can change the attributes of selected controls on a menu form in one operation.



(1) Components

In the Change as Batch dialog box, you can set the following items.

Text position

Select to change the text alignment of the selected controls.

This check box is disabled if you selected an edit control or combo box.

Vertical and Horizontal

From the drop-down lists, select the text position in the vertical and horizontal directions.

You can set the vertical and horizontal alignment only if you select the **Text position** check box.

Text color

Select to change the text color of all the selected controls, then specify the new color in the Set Color dialog box.

Background color

Select to change the background color of the selected controls, then specify the new background color in the Set Color dialog box.

Background

Select to change information about the background, then set the new information for the selected controls in the Properties (Background) dialog box

This check box is disabled if you select a control other than a static control, button or browse button.

Style

Select to change the style, then set information about the new style of the selected controls in the Properties (Style) dialog box.

This check box is available only if you selected controls of the same style.

You cannot change the **Default button** setting if you select a button or browse button.

You cannot change the **Command select field** setting if you select an edit control.

Font

Select to change the font in one operation, then specify the font information in the Font dialog box.

Font name

Shows the font currently selected.

Style

Shows the style of the selected font.

Size

Shows the size of the selected font.

Strikethrough

Shows whether to use strikethrough characters.

Underline

Shows whether to underline the text.

(2) Operations

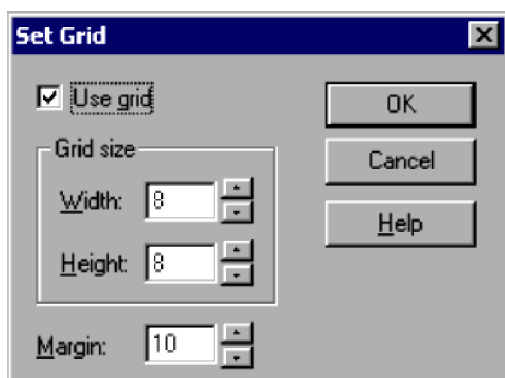
- Click **OK** to apply the settings and close the Change as Batch dialog box.
- Click **Cancel** to close the dialog box without changing any control attributes.

(3) Processing

- The initial value of each item is the value set for the control you selected first.
- The **OK** button is disabled when none of the check boxes is selected.

4.4.40 Set Grid dialog box

The Set Grid dialog box appears when you choose **Edit, Layout, Set Grid**. Set the grid spacing for display on a menu form.



(1) Components

In the Set Grid dialog box, you can set the following items.

Use grid

Select to display a grid.

Grid size

Width and Height

Specify in pixels the width and height of the grid.

Set values in the range 1 to 100. The default is **8**.

Margin

Specify in pixels the size of the margins.

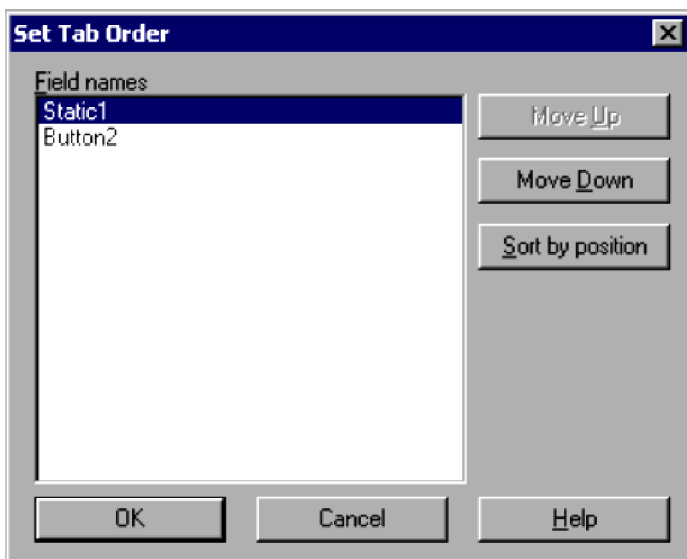
Set a value in the range 0 to 100. The default is **10**.

(2) Operations

- Click **OK** to apply the settings and close the Set Grid dialog box.
- Click **Cancel** to close the dialog box without applying the settings.

4.4.41 Set Tab Order dialog box

The Set Tab Order dialog box appears when you choose **Edit, Layout, Set Tab Order**. Set the order in which the tab key moves from one control to another.



(1) Components

In the Set Tab Order dialog box, you can set the following items.

Field names

Lists the names of the fields on the menu form. The tab key moves in sequence down the listed fields.

Move Up

Moves a selected field up one line in the tab order.

Move Down

Moves a selected field down one line in the tab order.

Sort by position

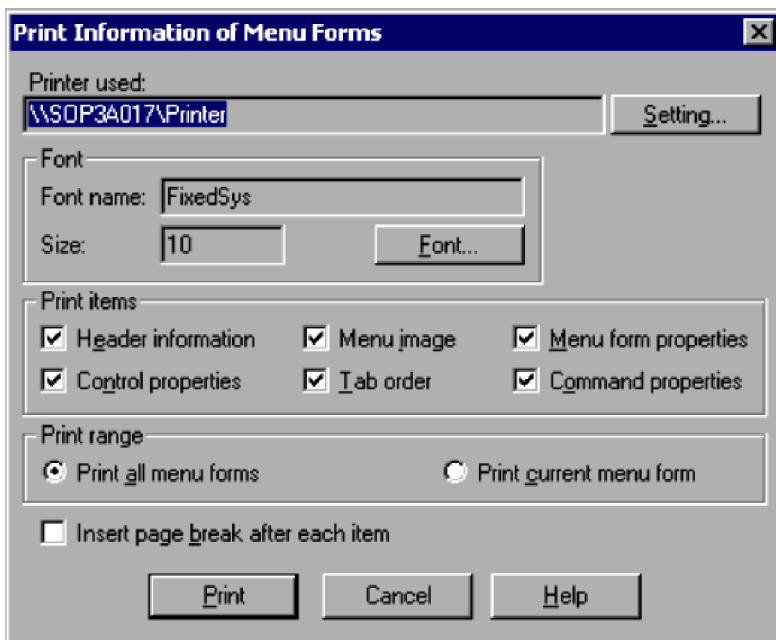
Sets the tab order from the top-left to bottom-right in the window, based on the field coordinates.

(2) Operations

- Click **OK** to apply the settings and close the Set Tab Order dialog box.
- Click **Cancel** to close the dialog box without changing the tab order.

4.4.42 Print Information of Menu Forms dialog box

The Print Information of Menu Forms dialog box appears when you choose **File, Print Menu Form**. Set preferences for printing the specifications of a completed menu form.



(1) Components

In the Print Information of Menu Forms dialog box, you can set the following items.

Printer used

Shows the printer to be used. To change the printer, click the **Setting** button.

The printer that is most often used is set by default.

Font name

Shows the print font name. To change the font, click the **Font** button.

FixedSys is set by default.

Size

Shows the print font size. To change the font size, click the **Font** button.

10 is set by default.

Print items

Header information

Prints header information. The following information is printed:

- Menu information file name
- Menu form name
- Date modified (*yyyy/mm/dd HH:MM:SS* format[#])
- Date printed (*yyyy/mm/dd HH:MM:SS* format[#])
- Page

[#]: *yyyy*: year; *mm*: month; *dd*: day; *HH*: hour; *MM*: minute; *SS*: second.

Menu image

Prints a text image of the menu form. The image is centered horizontally, and shrunk to fit the page if necessary.

Menu form properties

Prints the menu form properties.

Control properties

Prints control properties.

Tab order

Prints the tab order.

Command properties

Prints command properties.

All of the items are selected by default.

Print Range

Print all menu forms

Prints the specified information for all menu forms.

Print current menu form

Prints the specified information for the current menu form only.

If you have not selected a menu form, **Print current menu form** is disabled.

The **Print all menu forms** check box is selected by default.

Insert page break after each item

Prints each of the items that you selected in the **Print Items** field on a new page. This check box is unselected by default.

(2) Operations

- Click **Print** to print the specified information. The Print Information of Menu Forms dialog box closes when printing is finished.
- Click **Cancel** to close the dialog box without applying the settings.

(3) Processing

- The **File, Print Menu Form** command is disabled when there are no menu forms.

5

Troubleshooting

This chapter describes the types of problems that may occur in JP1/Script and how to handle them.

5.1 Troubleshooting procedure

This section describes the troubleshooting procedure in the event of a JP1/Script error.

Checking the event

In the event of an error, check the event. If a message has been issued, check the contents of the message.

For details about troubleshooting when JP1/Script is used, see [5.2 Troubleshooting](#). For details about the log information that is displayed by JP1/Script, see [5.3 Log information](#).

Acquiring information

To determine the cause of the error, you must acquire information. For details, see [5.4 Information to be acquired in the event of an error](#) and [5.5 How to acquire information](#).

Examining the error

Check the acquired information to determine the cause of the error and isolate the erroneous section or range.

5.2 Troubleshooting

This section describes troubleshooting when JP1/Script is used. If an error occurs while you are using JP1/Script, check for the events described in this section.

5.2.1 Troubleshooting script execution problems

Table 5-1 lists the problems you may encounter during script execution and the actions you should take.

Table 5–1: Troubleshooting script execution problems

Problem	Cause	Action
Unable during script execution to allocate space on the volume for trace storage or for command processing.	Insufficient space available on the volume.	Use JP1/Script Manager or Windows Explorer to free up space by deleting unnecessary files.
		Specify a different volume for the trace file output folder and work folder.
Script execution failed with message Initialization of application failed or termination code 99.	Insufficient memory to execute the script.	Add more memory or increase the page file size. If there is still insufficient memory after providing a larger page file size, quit other applications to increase the amount of available memory.
		If the script was executed from the JP1/Script service, change the service account to one that is not being used by any other service.
Script execution resulted in an error with message The input file is damaged or has a different file format.	The specified file is invalid.	Use a valid file or the correct file.
	If the script terminated abnormally during the previous run, the script file may not have been saved correctly.	Use JP1/Script Manager or Windows Explorer to delete the file.
Script file keeps on running.	The script was activated from a service program such as JP1/AJS, and a process requiring user intervention, such as an entry from a message box or window, was executed from within the script.	To start the script from a service program such as JP1/AJS and execute an interactive operation, use the <code>NetExec</code> command to execute the script in the logon space.
	A program called by the <code>Exec</code> or <code>NetExec</code> command went into a program loop.	Check the executable program for a programming error.
	The trace management file (<code>SPTLOGDB.SPB</code>) has been corrupted.	Use a program such as Windows Explorer to delete the trace management file.
When executed from a user program, the script terminated with code 19.	A syntax error occurred in the script.	Execute the script from JP1/Script Manager or Editor, and check the nature of the error.
		Make sure that you are not using an old version of script engine.
Script execution resulted in an error.	An error was detected during execution of the script file.	Start the JP1/Script Trace Viewer and check the nature of the error.
Script execution speed was slow.	Too many scripts were executed concurrently.	Reduce the number of scripts that are executing concurrently.
	The trace management file (<code>SPTLOGDB.SPB</code>) is too large.	Delete the trace management file (<code>SPTLOGDB.SPB</code>) while the script is not running and then re-execute the script. If

Problem	Cause	Action
Script execution speed was slow.	The trace management file (SPTLOGDB . SPB) is too large.	many user trace files with unique file names have been created by using the <code>Message</code> command, use the <code>TextOpen</code> , <code>TextWrite</code> , or <code>TextClose</code> command rather than the <code>Message</code> command to create trace files.
Script terminated abnormally due to insufficient memory (such as with code 20), or command execution resulted in an error due to insufficient memory (such as when the message <code>Insufficient memory is displayed</code> during execution of the <code>Copy</code> command).	The trace management file (SPTLOGDB . SPB) is too large.	Delete the trace management file (SPTLOGDB . SPB) while the script is not running and then re-execute the script. If many user trace files with unique file names have been created by using the <code>Message</code> command, use the <code>TextOpen</code> , <code>TextWrite</code> , or <code>TextClose</code> command rather than the <code>Message</code> command to create trace files.

5.2.2 Troubleshooting command execution problems

Table 5-2 describes the problems you may encounter during command execution and the actions you should take.

Table 5–2: Troubleshooting command execution problems

Problem	Cause	Action
When the <code>NetExec</code> command is executed, the message <code>The server does not respond to the connect request.</code> is displayed.	JP1/Script is not installed on the connection target computer.	Install JP1/Script on the computer you wish to connect to, and then restart Windows.
	The script was executed in the logon space, but the Script launcher or the Script Launcher service was not active on the connection target computer.	Start the Script launcher or the Script Launcher service.
	The script was executed in the service space, but the JP1/Script service was not active on the connection target computer.	Start the JP1/Script service.
	A logical host name (logical IP address) was specified as the connection target computer, but failover occurred at the logical host.	Specify a physical host name (physical IP address), not a logical host name (logical IP address), as the connection target computer. When you specify a logical host name as the connection target computer, register JP1/Script into the cluster.
When the <code>NetExec</code> command is executed, the message <code>unknown user name or bad password.</code> is displayed.	A script with <code>Logon</code> set as the start type was executed by using the automatic startup function or from JP1/Script Manager. If the script was executed directly, the user account used to log on to Windows does not exist on the connection target computer, or the <code>Guest</code> account is invalid.	Check the connection target computer to make sure that the <code>Guest</code> account is valid. Make sure that the user account used to log onto Windows exists on the connection target computer.
	A script whose start type is <code>Service</code> was executed using the automatic start function, but the user account set in the JP1/Script service does not exist on the connection target computer, or the <code>Guest</code> account is invalid.	Check the connection target computer to make sure that the <code>Guest</code> account is valid.
		Make sure that the user account set in the JP1/Script service exists on the connection target computer.

Problem	Cause	Action
When the <code>NetExec</code> command is executed, the message <code>unknown user name or bad password.</code> is displayed.	The script was started from a JP1/AJS service program, but the OS user account mapped to the JP1 user executing the script does not exist on the connection target computer, or the <code>Guest</code> account is invalid.	<p>Check the connection target computer to make sure that the <code>Guest</code> account is valid.</p> <p>Make sure that the OS user account mapped to the JP1 user executing the script exists on the connection target computer.</p>
	The script was started from a service program other than JP1/AJS, but the user account set in each service account does not exist on the connection target computer, or the <code>Guest</code> account is invalid.	<p>Check the connection target computer to make sure that the <code>Guest</code> account is valid.</p> <p>Make sure that the user account set in each service program exists on the connection target computer.</p>
When the <code>NetExec</code> command is executed, the message <code>The server is not logged on through the logon user specified in the startup parameter for the Script Launcher service.</code> is displayed.	The script was executed in the logon space of the Script Launcher service, but the user specified in the start parameter has not logged on.	Log on as the logon user specified in the start parameter.
When the <code>NetExec</code> command is executed, the message <code>The user who executed the NetExec command is not permitted to perform the operation in the call-destination computer.</code> is displayed.	The <code>NetExec</code> command was executed by a user who is not permitted to do so.	Log on as a user who is permitted to execute the <code>NetExec</code> command on the connection target computer.
When the <code>NetExec</code> command is executed, the message <code>Access is denied. Check the file attributes or the security.</code> is displayed.	The script was executed by using the automatic startup function or from a service program such as an upper-level service program, but a local system account was set as the system account.	If the script was executed by using the automatic startup function or from a service program such as an upper-level service, set a user account that exists on the connection target computer as the service account.
When the <code>SetGV</code> , <code>GetGV</code> , <code>DeleteGV</code> , or <code>NetExec</code> command was executed, the message <code>This command cannot be executed.</code> was displayed.	Settings at the connection target computer prohibit execution of this command on a client.	Check the JP1/Script Manager settings at the connection target computer to make sure that execution of this command is permitted on a client.
After the <code>Exec</code> , <code>NetExec</code> , or <code>CallSpt</code> command was executed, an unexpected code was set in the <code>_EXEC_RTN_</code> reserved variable.	An error occurred in the execution file that was started.	<p>Execute the executable file by itself to make sure that there is no error and that the executable file does not return the code that was set in <code>_EXEC_RTN_</code>.</p> <p>If an application error occurs in the executable file, a value such as <code>-1073741819</code> (hexadecimal <code>C0000005</code>) is set.</p>
When the <code>Beep</code> command was executed, an error code such as <code>0002</code> (the system cannot find a specified file) was output.	This is caused by a restriction of the OS or hardware.	Take corrective action, for example by using the <code>IMEventMessage</code> command. For details, see the notes in 8.13.3 Beep (sound a beep from the speakers) .
A beep did not sound when the <code>Beep</code> command was executed.		

5.2.3 Troubleshooting Trace Viewer problems

Table 5-3 describes the problems you may encounter when using Trace Viewer and the actions you should take.

Table 5–3: Troubleshooting Trace Viewer problems

Problem	Cause	Action
When connection is established with another computer from Trace Viewer, the message Trace Viewer server is not running or is not responding is displayed.	JP1/Script is not installed on the connection target computer.	Install JP1/Script on the connection target computer, and then restart Windows.
	The JP1/Script service has not started on the connection target computer.	Start the JP1/Script service on the connection target computer.
	A logical host name (logical IP address) was specified as the connection target computer, but failover occurred at the logical host.	Specify a physical host name (physical IP address), not a logical host name (logical IP address), as the connection target computer.
		If you specify a logical host name as the connection target computer, register JP1/Script into the cluster.

5.2.4 Troubleshooting Process Viewer problems

Table 5-4 describes the problems you may encounter when using Process Viewer and the actions you should take.

Table 5–4: Troubleshooting Process Viewer problems

Problem	Cause	Action
When connection is established with another computer from Process Viewer, the message The server does not respond to the connect request is displayed.	JP1/Script is not installed on the connection target computer.	Install JP1/Script on the connection target computer, and then restart Windows.
	The JP1/Script service has not started on the connection target computer.	Start the JP1/Script service on the connection target computer.
	A logical host name (logical IP address) was specified as the connection target computer, but failover occurred at the logical host.	Specify a physical host name (physical IP address), not a logical host name (logical IP address), as the connection target computer.
		If you specify a logical host name as the connection target computer, register JP1/Script to the cluster.
When connection is established with another computer from Process Viewer, the message The server returned rejection to the connect request is displayed.	The connection target computer is set to refuse requests from Process Viewer.	Check the connection target computer to make sure that the JP1/Script Manager is set to accept Process Viewer requests.

5.3 Log information

In the event of a JP1/Script error, check the log information and determine the action to be taken. JP1/Script outputs seven types of log information:

- **Analysis trace file**
Output only when output of an analysis trace file is specified in the execution environment settings.
- **Execution trace file**
Output only when output of an execution trace file is specified in the execution environment settings.
- **User trace file**
Output only when `Target_File` is specified in the first argument of the `Message` command.
- **Server trace file**
Output only when a remote computer is specified in the `SetGV`, `GetGV`, `DeleteGV`, or `NetExec` command.
- **NetExec error log file**
Output only when the `NetExec` command is executed.
- **Event log**
Output only when event log output is specified.
- **Maintenance log file**
Output only when an error was detected during execution of a script or in the JP1/Script service, JP1/Script Launcher service, or Script Launcher.

This section describes the seven types of log information.

5.3.1 Types of log information

(1) Analysis trace file

The *analysis trace file* is a type of log information used to output command analysis results.

For details about how to set this log information, see [4.1.12 Set Command Line dialog box](#) and [6.2 Rules for writing command lines](#). For details about the output formats, see [A. Output Formats of Script Trace Files](#).

(2) Execution trace file

The *execution trace file* is a type of log information used to output command execution results. The type of log information that is output to the execution trace file depends on the output level. For details about how to set this log information, see [4.1.12 Set Command Line dialog box](#) and [6.2 Rules for writing command lines](#). For details about the output formats, see [A. Output Formats of Script Trace Files](#).

(3) User trace file

The *user trace file* is a type of log information that is output when `Target_File` is specified in the first argument of the `Message` command. For details about how to set this log information, see [8.6.2 Message \(output text to a file or window\)](#). For details about the output formats, see [A. Output Formats of Script Trace Files](#).

(4) Server trace file

The *server trace file* is a type of log information used to output the results of command execution on the server when the name of another computer is specified in the SetGV, GetGV, DeleteGV, and NetExec commands. For details about how to set this log information, see [4.1.22 Options \(Server Information\) dialog box](#). For details about the output formats, see [A. Output Formats of Script Trace Files](#).

(5) NetExec error log file

The *NetExec error log file* is a type of log information used to output errors that occur during execution of the NetExec command. For details about the NetExec error log file, see [A. Output Formats of Script Trace Files](#).

(6) Event log

The *event log* is a type of log information used to report the system status and errors. You can use Windows Event Viewer to view the contents. For details about how to set this log information, see [4.1.12 Set Command Line dialog box](#) and [6.2 Rules for writing command lines](#).

(7) Maintenance log file

The *maintenance log file* is used to record detailed information about the errors that occur when Windows functions are called in JP1/Script programs. If you cannot determine the cause of the error by referring to trace files, log files, and event logs and need to contact the support center, send the maintenance log file. For details about how to enable the maintenance log file, see [D. Maintenance Log Files](#). The format of output information is not made public.

5.3.2 List of log files and directories

Table 5-5 describes the log output sources, log file names, and required disk space for the log information that is output from JP1/Script.

Table 5–5: Log output sources, log file names, and required disk space

Type of log information	Log file name	Required disk space (KB)
Analysis trace file	<i>script-execution-folder</i> ^{#1} \ <i>script-file-name</i> .SPA	154 ^{#2}
Execution trace file	<i>script-execution-folder</i> ^{#1} \ <i>script-file-name</i> .SPX	154 ^{#2}
User trace file	File path specified in Message command's OutputName	154 ^{#2}
Server trace file	<i>installation-folder</i> ^{#3} \Data ^{#4} \SPTSVTRC.SPY	204
NetExec error log file	<i>installation-folder</i> ^{#3} \Log\STXNetExec_Client\ ^{#5} STXNetExec_C_xx ^{#6} .log	10 ^{#7}
	<i>installation-folder</i> ^{#3} \Log\STXNetExec_Server\ ^{#5} STXNetExec_S_xx ^{#6} .log	10 ^{#7}
Maintenance log file	<i>installation-folder</i> ^{#3} \Log\Maintenance\ ^{#8} XXXX_ <i>process-ID</i> .log ^{#9}	4,032 ^{#10}
Program execution information management file	<i>installation-folder</i> ^{#3} \Log\hliclibtrc.conf	0.25
	<i>installation-folder</i> ^{#3} \Log\hlicliberr.conf	0.25

Type of log information	Log file name	Required disk space (KB)
Program execution information management file	<i>installation-folder</i> ^{#3} \Log\hliclibmgrtrc.conf	0.25
	<i>installation-folder</i> ^{#3} \Log\hliclibmgrerr.conf	0.25
Program execution information file	<i>installation-folder</i> ^{#3} \Log\hliclibtrcX.log ^{#11}	5,120 ^{#12}
	<i>installation-folder</i> ^{#3} \Log\hlicliberrX.log ^{#11}	5,120 ^{#12}
	<i>installation-folder</i> ^{#3} \Log\hliclibmgrtrcX.log ^{#11}	5,120 ^{#12}
	<i>installation-folder</i> ^{#3} \Log\hliclibmgrerrX.log ^{#11}	5,120 ^{#12}

#1

You can change the output target folder. For details about how to set this log information, see [4.1.10 Set Execution Environment \(Trace Information\) dialog box](#) or [4.1.25 Options \(Multi-activation\) dialog box](#).

#2

This is the default value. For details about the required disk space, see [1.4.2 File sizes](#).

#3

This folder is created in the \ProgramData\Hitachi\Script folder on the system drive.

#4

You can change the output target folder. For details about how to set this log information, see [4.1.28 Options \(Cluster Environment\) dialog box](#).

#5

You can change the output target folder. For details about how to set this log information, see [8.10.2 NetExec \(call an executable file on the local PC or remote PC\)](#). The STXNetExec_Client folder is created at the client where the NetExec was executed, while the STXNetExec_Server folder is created on the server computer specified in the NetExec command.

#6

xx is a numeric value in the range 01 to 30.

#7

This is the maximum value for a single file. Because a maximum of 30 files are created, the maximum required size is about 300 kilobytes.

#8

You can change the output target folder. For details about how to set up to log information, see [D. Maintenance Log Files](#).

#9

The file name depends on the program that outputs the maintenance log file. For details about file names, see [D. Maintenance Log Files](#).

#10

This is the maximum value when a file is output with the default value. For details about file sizes, see [D. Maintenance Log Files](#).

#11

The last character X in the log file name not including its file extension is a number in the range from 1 to 5. A maximum of five files are created, and log entries are recorded by using a wrap-around principle. Each file can reach a maximum of 1,024 kilobytes.

#12

This is the maximum disk space required when five files are created.

5.4 Information to be acquired in the event of an error

If an error cannot be corrected by the action described in [5.2 Troubleshooting](#), you should acquire information that will be useful in determining the cause of the error, and then you should notify the system administrator. This section describes the information to be acquired in the event of an error.

5.4.1 OS information

Table 5-6 describes the OS information to be acquired in the event of an error.

Table 5–6: OS information to be acquired in the event of an error

Type of information	Overview	Acquisition method
Event log (application log)	Windows event log	Save the log file from Event Viewer.
Event log (system log)		
Dump information ^{#1}	Process dump ^{#2}	See Microsoft(R) support technical information.

#1

Acquire this information in the event of an application error.

#2

Acquire this information when the script process (`SPTXE.EXE`) keeps on running.

5.4.2 JP1/Script information

You should acquire the JP1/Script-related information that is described below. In the event of a network connection error, you should also acquire files on the connection target machine.

Table 5-7 describes the JP1/Script information to be acquired in the event of an error.

Table 5–7: JP1/Script information to be acquired in the event of an error

Type of information	Overview	Storage location
Script file	Script source during script execution	<i>script-execution-folder\script-file-name.SPT</i>
Script execution environment file	File containing the environment settings during script execution	<i>script-execution-folder\script-file-name.SPV</i>
Monitoring information file	File containing monitoring information	<i>script-execution-folder\script-file-name.SPD</i>
Menu file	Menu file during	<i>menu-file-name.SPN</i>

Type of information	Overview	Storage location
Menu file	execution of Menu command	<i>menu-file-name</i> .SPN
Analysis trace file	Analysis trace file during script execution	<i>script-execution-folder\script-file-name</i> .SPA
Execution trace file	Execution trace file during script execution	<i>script-execution-folder\script-file-name</i> .SPX
User trace file	Trace file during execution of Message command	File path specified in Message command's OutputName
Server trace file	Execution results log on the server that is output when another computer is specified in the NetExec, SetGV, GetGV, or DeleteGV command	<i>installation-folder</i> #\Data\SPTSVTRC.SPY
NetExec error log file	Error information that is output to client and server during execution of NetExec command	<i>installation-folder</i> #\Log\STXNetExec_Client\STXNetExec_C_xx.log
		<i>installation-folder</i> #\Log\STXNetExec_Server\STXNetExec_S_xx.log
Automatic start information file	File containing automatic start information	Shared by all users: <i>system-drive</i> \ProgramData\Hitachi\Script\Data\SPTLNCH.SPH For a specific user: <i>system-drive</i> \ProgramData\Hitachi\Script\Data\ <i>current-user-name</i> \SPTLNCH.SPH
Server environment file	File containing server environment	<i>installation-folder</i> #\Data\SPTS.V.SPS
Trace management file	File for managing trace files	<i>installation-folder</i> #\Data\SPTLOGDB.SPB
Global variables file	File containing global variables	<i>installation-folder</i> #\Data\SPTGV.SPG

Type of information	Overview	Storage location
Registry information	Registry information for JP1/Script	Under HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\
Maintenance log file	Detailed information about the errors, including how they occurred	<i>installation-folder</i> #\Log\Maintenance\XXXX_process-ID.log
Other	ini files handled by script processing, log files created during script processing, etc.	--

This folder is created in the \ProgramData\Hitachi\Script folder on the system drive.

5.4.3 Details of operation

In the event of an error, it is essential that you note the following operation-related information:

- Details of the operation that was underway
- Time the error occurred
- Machine configuration (such as each OS version and host name)
- Replicability of the error

5.4.4 Information on the screen

Obtain a hardcopy of the following screen displays:

- In the event of an application error, a hardcopy of the window operation
- Hardcopy of the error message dialog box (entire screen)
- Hardcopy of Process Viewer (entire screen)
- Hardcopy of the Task Manager's SPTXE process (also acquire the number of threads by choosing **View**, **Select Column**, and then selecting the **Number of threads** option)

5.5 How to acquire information

This section explains how to acquire information in the event of an error.

5.5.1 Data collection tool

(1) Preparing the data collection tool

The data collection tool is the `Sptras` file stored in the `Tools\EngUS` folder in the installation folder. Use Explorer or its equivalent to copy the `Sptras` file to any folder. Then, change the extension of the copy of the file to `.bat`. Note that you need to edit the references to the installation folder and `Data` folder in the `Sptras.bat` file by using an editor according to the instructions in the `Sptras` file. Do not, however, enclose the folders with double quotation marks (`"`).

If you have changed the log output destination or other settings, you must also change the contents of the `Sptras` file according to the instructions in the `Sptras` file. Again, do not enclose the folders with double quotation marks (`"`).

(2) Executing the data collection tool

Execute the data collection tool as follows:

```
c:\>c:\temp\Sptras.bat
```

The execution results of the data collection tool are output to `%TEMP%\jplscript` by default. Collect the information in this folder. If a file containing the execution results of the batch file already exists, a message asking you whether you want to overwrite the information appears. Enter `y` in response to the message.

Important note

Before you execute the batch file of the data collection tool, use Script Viewer to confirm that no script file is being executed. In addition, do not execute any scripts until the data collection tool terminates.

(3) Information acquired by the data collection tool

The following table shows the information acquired by the data collection tool and lists the names of the folders or files to which the information is output.

Table 5–8: Information acquired by the data collection tool and output destinations

Information collected by the data collection tool	Information is output to:
List of files in the JP1/Script installation folder	<code>FILELIST.txt</code>
<code>hosts</code> file	<code>hosts</code>
<code>services</code> file	<code>services</code>
JP1/Script registry information	<code>REGDATA.DAT</code>
OS version, Windows environment variable information, and IP configuration information	<code>OSINFO.TXT</code>
All information in the JP1/Script installation folder	Backup (folder)

Information collected by the data collection tool	Information is output to:
Maintenance log file	Backup\MaintenanceLog (folder)

5.5.2 Dump information

In the event of an application error, acquire the dump information described in [5.4.1 OS information](#).

5.5.3 Acquiring JP1/Script information

If a problem occurred, acquire the following files:

- The script file (extension: SPT) in which the problem occurred, and the execution environment file (extension: SPV) of that script file
- Monitoring information file (extension: SPD)
- Menu file (extension: SPN)
- Analysis trace file (extension: SPA)
- Execution trace file (extension: SPX)
- User trace file (output by the `Message` command)
- Files processed in a script file (such as the `ini` file or a file specified in the `TextOpen` command)
- Maintenance log file (extension: LOG)

5.5.4 Acquiring Windows event logs

In the Windows Event Viewer window, save the Windows event log (application log or system log) as an event log file or as a text file.

5.6 Backup and recovery

This section describes backup and recovery of the files and operating environment information that are used by JP1/Script.

5.6.1 Backing up and recovering the files used by JP1/Script

Table 5-9 shows the files to be backed up. The user must back up and recover the files if necessary. Before you start the backup process, stop the JP1/Script service, Script Launcher, and the Script Launcher service. If any of these services is running during backup of files or folders, a lock error might occur.

Note that the files other than the script file might not exist because they are created after a script is defined or executed.

Table 5–9: Files to be backed up

No.	File name	File name or extension	File location
1	Script file	.SPT	Created by the user in any folder.
2	Execution environment file	.SPV	Created in the script execution folder after a script has been defined or executed.
3	Menu information file	.SPN	
4	Monitoring information file	.SPD	
5	Analysis trace file	.SPA	
6	Execution trace file	.SPX	
7	User trace file	.TXT, etc.	File path specified in <i>OutputName</i> of the Message command
8	Server trace file	SPTSVTRC.SPY	Created in the <i>installation-folder</i> \Data folder ^{#1, #2} after a script has been defined or executed
9	Trace management file	SPTLOGDB.SPB	
10	Server environment file	SPTSV.SPS	
11	Global variable file	SPTGV.SPG	
12	Automatic start information file ^{#1}	SPTLNCH.SPH	Created in the <i>installation-folder</i> \Data folder ^{#1, #3} after a script has been defined or executed
13	Execution environment syntax file	.SPU	Created in the same folder as for the execution environment file, or in the folder specified in <i>toDirName</i> . This file exists only when Execution Environment File Converter is used.

#1

If you change the output destination for management files by selecting **Tools, Options**, and then **Cluster Environment** in the Script Manager window, the server trace file is created in the new destination folder.

#2

These files are created in the *system-drive*\ProgramData\Hitachi\Script\Data folder.

#3

These files are also created in the *system-drive*\ProgramData\Hitachi\Script\Data folder and its subfolder. For details, see [5.4.2 JP1/Script information](#).

5.6.2 Backing up and recovering operating environment information

- Information set using the **Tools, Options** command of Manager

Except for settings on the **Server Information** page, the settings in the Options dialog box, which is displayed by choosing **Tools** and then **Options** in the Script Manager window, are written to the registry. You can simply make note of the settings, then re-enter them in the dialog box at the time of recovery.

- JP1/Script service account

To recover the backed up information, make note of the account of the JP1/Script service and set the information again.

- Other information

At the time of recovery, simply re-enter the applicable values set in the registry.

6

JP1/Script Coding Conventions

In JP1/Script, you can create scripts using statements, basic commands, and special commands. This chapter describes the rules relating to scripts and command lines.

6.1 Rules for creating scripts

This section describes the following coding conventions that you need to know when creating a script:

- Variable naming conventions
- Maximum number and data size of variables
- Restricted keywords
- Reserved variables
- Array variables
- Constants
- Numeric coding conventions
- String coding conventions
- Operation conventions
- Operator precedence
- Script coding conventions
- JP1/Script exit codes
- JP1/Script event logs

6.1.1 Variable naming conventions

JP1/Script uses the following variable naming conventions:

- The first character in a variable name must be an alphabetic (A to Z, and a to z). For the other characters, alphabets, numerics, and underscores may be specified.
- A variable name must not exceed 32 bytes. If a variable name contains more than 32 bytes, the 33rd and subsequent bytes are ignored. If a two-byte character is specified for the 32nd byte, the 32nd byte is also ignored.
- Variable names and keywords are not case sensitive.
- Variable names must not span two or more lines of coding.
- Keywords must not be used as variable names.
- A variable name may be repeated inside a variable declaration block.
- Procedure[#] names are subject to the same naming conventions as variables. A procedure and a variable must not have the same name.

#

A procedure is a set of commands processed at run time as a single unit. In JP1/Script, you can use the `Function` statement or `Sub` statement to define a procedure. The term *sub-routine* is synonymous with procedure.

6.1.2 Maximum number and data size of variables

The following restrictions apply to the number and data size of variables used in JP1/Script:

- Up to 1,024 variables can be used in one procedure.

- Up to 1,024 variables can be used outside a procedure (excluding reserved variables and location variables).
- The maximum data size that can be stored in a variable is 1,024 bytes. Data exceeding this maximum size is ignored.

6.1.3 Restricted keywords

Table 6-1 lists the keywords (JP1/Script reserved words) that must not be used as variable names.

Table 6–1: Restricted keywords

Letter	Restricted keywords not permitted as variable names
A	Abort, AbortAll, AbortRetryIgnore, Abs [#] , AddStr, After, Alert, Alertness, AllDq, AllGV, Alt, And, Anyway, Append, AppliModal, ApplicationModal, Array [#] , Asc [#] , AscB [#] , AscW [#] , Atn [#] , ATTR_ARCHIVE, ATTR_COMPRESSED, ATTR_HIDDEN, ATTR_NORMAL, ATTR_OFF, ATTR_OFFLINE, ATTR_ON, ATTR_READONLY, ATTR_SUBDIR, ATTR_SYSTEM, ATTR_TEMPORARY, AuditFailure, AuditSuccess
B	Backup, Beep, Before, BitmapHide, BitmapShow, Boolean [#] , ByRef [#] , Byte, ByVal [#]
C	CalcDate, CalcTime, Call, CallDll, CallSpt, Cancel, CancelStartUp, CancelUserErr, Case, CatFiles, CBool [#] , CByte [#] , CDate [#] , CDb1 [#] , CDROM, CheckDirName, CheckDriveType, Chr [#] , ChrB [#] , ChrW [#] , Cint [#] , Clear [#] , CLng [#] , Close, Command, CompDate, CompTime, Continue, Copy, Cos [#] , Create, CreateObject [#] , Critical, CSng [#] , CStr [#] , Ctrl, Ctrl_Alt, Currency [#] , Cur_Desktop, Cur_Program, Cur_Startmenu, Cur_Startup
D	Date, DateSerial [#] , DateValue [#] , Delete, DeleteGV, Day, DayU, Debug, DeleteDir, DeleteFile, DeleteGroup, DeleteShortcut, DependG, DependM, Description [#] , Destroy, Dim, DISABLE, DispName, Do, Double [#]
E	Each [#] , Else, ElseIf, Emergence, Emergency, Empty, End, Enter, EntryStartUp, Equal, Eqv [#] , Erase [#] , Err, Errctl, Error, ErrSkip, ErrSkip2, Esc, Ex, Exclamation, ExclDir, Exec, EXEC_RUNNING, EXEC_STOPPED, Exit, ExitWindows, Exp [#] , Explicit [#]
F	False, FileTime, Fix [#] , FIXED, For, Force, Format, FreeExt, Function, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12
G	GetArrayCount, GetDateCount, GetDiskFreeSpace, GetEnv, GetEnvironment, GetErrorMessage, GetExecStatus, GetFileAttr, GetFileAttribute, GetFileSize, GetFileTime, GetGV, GetPath, GetProcessCount, GetProcessInfo, GetServiceName, GetTextPosition, GetTimeCount, GetVerInfo, GetVersionInfo, GetVolLabel, GetVolumeLabel, Goto, Group
H	HelpContext [#] , HelpFile [#] , Hex [#] , HIDE, HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, Hour, HourU
I	If, IfEmpty, Ignore, IMEventMessage, Imp [#] , InArray, Info, Information, IniRead, IniWrite, InputBox, InputLine, InStr, InStrB [#] , Int [#] , Integer [#] , Is [#] , IsArray [#] , IsDate [#] , IsDef, IsDefine, IsEmpty, IsEmptyDir, IsEmptyGroup, IsEmptyReg, IsExistDir, IsExistFile, IsExistRegKey, IsExistService, IsFileAttr, IsFileAttribute, IsLeapYear, IsLower, IsMultiChar, IsNew, IsNull [#] , IsNumeric, IsObject [#] , IsSingleChar, IsUpper, IsWriteableDir
K	KB
L	LBound [#] , LCase, Lcl_Desktop, Lcl_Program, Lcl_Startmenu, Lcl_Startup, Left, LeftB [#] , Len, LenB [#] , Log [#] , Logoff, Logon, LOGON_FAILED, Long [#] , Loop, Ltrim
M	MakeDir, MakeGroup, MakePath, MakeShortcut, Max, MB, Menu, Message, MessageBox, MessageEventLog, Mid, MidB [#] , Min, Minus, Minute, MinuteU, Mod, Mod=, Modify, Month, MonthU, Move

Letter	Restricted keywords not permitted as variable names
N	Name, NeedDq, NetExec, Next, No, NOCHANGE, None, NoOverwrite, NoReplace, Normal, Not, NotEqual, Nothing#, Notice, Now#, Null#, Number#
O	Object#, Oct#, OK, OKCancel, On, Option#, Or, Overwrite, OverwriteOnly
P	Password, Path, Pause, File, Plus, Poweroff, Preserve#, Private#, ProcessEnv, Public#
Q	Question
R	Raise#, RAMDISK, Randomize#, ReadOnly, ReadWrite, Reboot, ReDim#, REG_BINARY, RegDelete, RegDeleteKey, REG_DWORD, REG_DWORD_BIG_ENDIAN, REG_EXPAND_SZ, REG_LINK, REG_MULTI_SZ, REG_NONE, RegRead, REG_RESOURCE_LIST, REG_SZ, RegWrite, REMOTE, REMOVABLE, Release, Rem, Remove, Rename, Replace, ResetStandardFile, ResetStdFile, Restart, Resume#, Retry, RetryCancel, Right, RightB#, Rnd#, Rtrim, ResetRetryMode, ResetTrialOpenMode
S	Second, SecondU, Security, Select, SplitFile, SeparateStr, SeparateStrCount, Service, ServiceChange, ServiceContinue, SERVICE_AUTO_START, SERVICE_BOOT_START, SERVICE_CONTINUE_PENDING, ServiceControl, SERVICE_CONTROL_CONTINUE, SERVICE_CONTROL_PAUSE, SERVICE_CONTROL_STOP, ServiceCreate, ServiceDelete, SERVICE_DEMAND_START, SERVICE_DISABLED, SERVICE_ERROR_CRITICAL, SERVICE_ERROR_IGNORE, SERVICE_ERROR_NORMAL, SERVICE_ERROR_SEVERE, SERVICE_FILE_SYSTEM_DRIVER, ServiceGetValue, SERVICE_KERNEL_DRIVER, ServicePause, SERVICE_PAUSE, SERVICE_PAUSE_PENDING, ServiceQuery, ServiceRefer, SERVICE_RUNNING, ServiceSetValue, ServiceStart, SERVICE_START_PENDING, ServiceStop, SERVICE_STOPPED, SERVICE_STOP_PENDING, SERVICE_SYSTEM_START, SERVICE_WIN32_OWN_PROCESS, SERVICE_WIN32_SHARE_PROCESS, Set#, SetEnv, SetEnvironment, SetFileAttr, SetFileAttribute, SetFileTime, SetGV, SetPath, SetStandardFile, SetStdFile, SetVolLabel, Shutdown, SetVolumeLabel, Shift, Shift_Alt, Shift_Ctrl, Shift_Ctrl_Alt, Sgn#, Sin#, Single#, Skip, Sleep, Source#, Space, SplitPath, Sqr#, Start, StartName, StdError, StdInput, StdOutput, Step, Stop, Str, StrComp#, String, StringJ, Sub, SubDirToo, Submit, Syntax#, Sysmodal#, SystemEnv, SystemModal, SetRetryMode, SetTrialOpenMode
T	Tan#, Target_DispClear, Target_DispOff, Target_DispOn, Target_File, Target_SPAMFile, Target_SPXFile, TempDir, TempFile, TerminateProcess, TextClose, TextFileReplace, TextOnly, TextOpen, TextRead, TextSeek, TextWrite, Then, Time, TimeSerial#, TimeValue#, To, ToBegin, ToEnd, Trace, Trim, True, Twice, Type, TypeOf#
U	UBound#, UCase, UnSubmit, Until, Update, UserEnv, UserErr
V	Val#, Variant#, VarType#, Version, VersionUp
W	Wait, WaitAll, WaitForExec, Warning, Weekday, Wend#, While, WriteOnly
X	Xor#
Y	Year, Yes, YesNo, YesNoCancel
Symbol	^, ^=, -, -=, *, *=, /, /=, \, \=, +, +=, &, &=, ?, =, <>, <, >, <=, >=

#

These keywords will be progressively supported in later enhancements and have been allocated as keywords in advance. Do not use these keywords in variable names.

6.1.4 Reserved variables

JP1/Script provides a number of reserved variables for referencing specific data (system information and command return values).

Table 6-2 shows the various categories of reserved variables and their meaning.

Table 6–2: Reserved variables

Category	Reserved variable	Meaning
System reserved variable	<code>_ALLRIGHT_</code>	YES when the user has administrator privilege; Empty without administrator privilege.
	<code>_BIN_</code>	Name of the work folder for the script file being executed. Followed by a backslash (\).
	<code>_COMP_</code>	Name of the computer on which the script file is being executed.
	<code>_DOMAIN_</code>	Name of domain on which the user is logged.
	<code>_OS_</code>	Operating system and version: <ul style="list-style-type: none"> • For Windows 7 or Windows Server 2008 R2: WIN_NT6.1 • For Windows Server 2012 or Windows 8: WIN_NT6.2 • For Windows Server 2012 R2 or Windows 8.1: WIN_NT6.3 • For Windows 10: WIN_NT10.0
	<code>_OS_PLATFORM_</code>	Operating system: <ul style="list-style-type: none"> • WIN_NT
	<code>_OS_REVISION_</code>	Operating system minor version: <ul style="list-style-type: none"> • For Windows 7 or Windows Server 2008 R2: 1 • For Windows Server 2012 or Windows 8: 2 • For Windows Server 2012 R2 or Windows 8.1: 3 • For Windows 10: 0
	<code>_OS_VERSION_</code>	Operating system major version: <ul style="list-style-type: none"> • For Windows 7, Windows Server 2008 R2, Windows Server 2012, Windows 8, Windows Server 2012 R2, or Windows 8.1: 6 • For Windows 10: 10
	<code>_SCF_</code>	Folder name of the script file being executed. Followed by a backslash (\).
	<code>_SCF_FIL_</code>	File name of the script file being executed. No extension.
	<code>_SCF_EXT_</code>	Extension of a script file (.SPT).
	<code>_SNF_EXT_</code>	Extension of a menu information file (.SPN).
	<code>_SVF_EXT_</code>	Extension of an execution environment file (.SPV).
	<code>_SDF_EXT_</code>	Extension of a monitoring information file (.SPD).
	<code>_SYS_</code>	MS-DOS startup folder name. Followed by a backslash (\).
<code>_TEMP_</code>	Folder name for temporary files. Followed by a backslash (\).	
<code>_USER_</code>	User name of the person logged onto the system.	
<code>_WIN_</code>	Windows folder name. Followed by a backslash (\).	

Category	Reserved variable	Meaning
System reserved variable	<code>_WINSYS_</code>	Folder name in which the Windows driver resides. Followed by a backslash (\).
Process reserved variable	<code>_PROC_ID_</code>	Process identifier of the script being executed.
	<code>_ARGV_</code>	Array variable that stores location variables from %1, accessed in the form <code>_ARGV_(n)</code> where <i>n</i> is any numeric starting at 1.
	<code>_ARGV_CNT_</code>	Total number of location variables from %1. %0 is not included.
Command return value reserved variable	<code>_COPY_RTN_</code>	Execution result of the <code>Copy</code> command.
	<code>_COPY_CNT_</code>	Number of files copied by the <code>Copy</code> command.
	<code>_COPY_SKIP_CNT_</code>	Number of files not copied by the <code>Copy</code> command.
	<code>_COPY_SKIP2_CNT_</code>	Number of files skipped with <code>ErrSkip2</code> specified in the <code>Copy</code> command.
	<code>_EXEC_RTN_</code>	Return value of the <code>Exec</code> , <code>NetExec</code> , or <code>CallSpt</code> command. Signed numeric.
	<code>_EXEC_ID_</code>	Executable file ID for the <code>Exec</code> or <code>NetExec</code> command. Valid only when <code>Flag</code> is set to <code>False</code> (do not wait for called program to exit).
	<code>_JOB_RTN_</code>	Return value of a job operating command.
	<code>_MSG_RTN_</code>	Return value of the <code>InputDialog</code> or <code>MessageBox</code> command.
	<code>_DLL_RTN_</code>	Return value of an external function of the <code>CallDll</code> command.
	<code>_FORM_TERM_KEY_</code>	Reason for termination of a window (menu form).
	<code>_FORM_TERM_CMDNO_</code>	Command number when a window (menu form) is terminated by command execution.
	<code>_FORM_MODIFY_KEY_</code>	Modifier key of the window (menu form) termination key.
	<code>_FORM_FIELD_NAME_</code>	Name of the field that last had the focus.
	<code>_RTN_</code>	Error detail code. Signed numeric. Execution results of a command are set. However, for a command that returns a specific value (such as a character string manipulation command), 0 is always set in the <code>_RTN_</code> reserved variable. The cause of the error indicated in the error detail code will be output to the trace file as an error message.
	<code>_RTNxx_</code>	Return value of the <code>CallDll</code> command (where <i>xx</i> is a number from 00).
<code>_SVC_RTN_</code>	Return value of a service operating command.	
Character code reserved variable	<code>_NL_</code>	New line character.
	<code>_TAB_</code>	Tab character.
Error detail code reserved variable	<code>_NO_ERR_</code>	No error.
	<code>_ERR_EOF_</code>	End of file.

Category	Reserved variable	Meaning
Error detail code reserved variable	<code>_ERR_TIMEOUT_</code>	Timeout exceeded.
	<code>_ERR_FILE_</code>	File not found.
	<code>_ERR_PATH_</code>	Path not found.
	<code>_ERR_ACCESS_</code>	Access denied.
	<code>_ERR_PROTECT_</code>	Write-protected.
	<code>_ERR_READY_</code>	Device not ready.
	<code>_ERR_EXCLUSIVE_</code>	Another process is accessing the file.
	<code>_ERR_SVR_CONNECT_</code>	Server does not respond to the connection request.
	<code>_ERR_SVR_TIMEOUT_</code>	Timeout occurred while waiting for a response from the server.
	<code>_ERR_SVR_RECEIVEDATA_</code>	Error occurred while receiving data from the server.
	<code>_ERR_SVR_NODATA_</code>	Allowed time for no data transmission was exceeded during communication with the server.
	<code>_ERR_FILE_SIZE_</code>	Value cannot be set in the variable because the acquired value exceeds the variable's maximum value.
	<code>_ERR_NOT_LARGE_FILE_</code>	The size of the specified file is greater than the maximum value.
	<code>_ERR_FILE_POSITION_</code>	Read/write start location is beyond 2,147,483,647.
<code>_ERR_SERVICE_NOT_BEGIN_</code>	The JP1/Script service is not running.	

6.1.5 Array variables

Index numbers for array variables start at 1. An index number for a one-dimensional variable represents an element. A two-dimensional variable has two index numbers; the first represents the row element, and the second represents the column element.

The following shows the data structure of a two-dimensional array variable $T(5, 6)$.

$T(1, 1)$	$T(1, 2)$	$T(1, 3)$	$T(1, 4)$	$T(1, 5)$	$T(1, 6)$
$T(2, 1)$	$T(2, 2)$	$T(2, 3)$	$T(2, 4)$	$T(2, 5)$	$T(2, 6)$
$T(3, 1)$	$T(3, 2)$	$T(3, 3)$	$T(3, 4)$	$T(3, 5)$	$T(3, 6)$
$T(4, 1)$	$T(4, 2)$	$T(4, 3)$	$T(4, 4)$	$T(4, 5)$	$T(4, 6)$
$T(5, 1)$	$T(5, 2)$	$T(5, 3)$	$T(5, 4)$	$T(5, 5)$	$T(5, 6)$

(1) Array variable coding conventions

The following coding conventions apply to array variables:

- Array variable names are subject to the same naming conventions as variables.
- Index numbers representing the elements of an array variable (or row and column elements of a two-dimensional array) start at 1. Enclose index numbers in parentheses or square brackets.

- An array variable declared with an index number has a fixed number of elements. An array variable declared without an index number has a variable number of elements.
- An array variable may have one or two dimensions.
- The maximum number of elements is 65,536 for an array variable, and 131,072 for a script file.
- Once declared, the dimensions of an array variable must not be changed.

Example:

```
Dim A(5)
Dim A(5,10) #
```

#

(Error) You cannot change the number of dimensions.

- The number of elements in an array variable can be changed as long as the declared dimensions remain unchanged. An array variable with a fixed number of elements can be changed to a dynamic array variable with a variable number of elements, and vice versa. If the number of elements is changed, the previous setting value is set to the Empty value.

Example:

```
Dim A(10)
Dim A(5) #1
Dim A( ) #2
```

#1

You can change the number of elements.

#2

You can change the number to a variable number of elements.

- A non-array variable cannot be redefined as an array variable. An array variable cannot be redefined as a non-array variable.

Example:

```
Dim A
Dim A( ) #
```

#

(Error) You cannot redefine A because it is defined as a non-array variable.

- You can assign values to all the elements of an array variable by using one assignment statement. However, you cannot do so if the array variable on the left has a fixed number of elements and the number of elements varies between the array variable on the left and the array variable on the right.

Example 1:

```
Dim A(5), B( ), C(10)
For cnt = 1 To 5
    A(cnt) = Time
Next
B = A #1
C = B #2
```

#1

You can assign all the elements of A to B in one assignment statement. After the assignment, B has five elements.

#2

(Error) You cannot assign all the elements of B to C in one assignment statement because B and C have a different number of elements (5 versus 10).

Example 2:

```
Dim A( ), B(1)
For cnt = 1 To 5
    A(cnt) = Time
Next
B(1) = Date
A = B#
```

#

You can assign all the elements of B to A in one assignment statement. After the assignment, A has one element.

Example 3:

```
Dim A( ), B( )
For cnt = 1 To 5
    A(cnt) = Time
Next
For cnt = 1 To 10
    B(cnt) = Date
Next
A = B#
```

#

You can assign all the elements of B to A in one assignment statement. After the assignment, A has 10 elements.

- All the elements of a single row can be assigned in one assignment statement, but it is not possible to assign all the elements of a single column in one assignment statement.

Example:

```
Dim A(2,5), B( )
For cnt = 1 To 5
    A(1,cnt) = Time
Next
B = A(1)#
```

#

You can assign the elements of the first row of A to B in one assignment statement.

- Equality comparisons can be made on the elements of two array variables, but comparisons to determine which array variable is greater are not allowed.

Example 1:

```
Dim A(5), B(5)
...
If A = B Then#
```

#

You can compare A and B to determine whether they are equal.

Example 2:


```
Dim A(5), B(5)
...
If A < B Then#
```


 (Error) You cannot compare A and B to determine which is greater.

- When an intermediate element in a dynamic array variable is undefined, Empty values are assigned from the first element to the undefined element. Note, however, that the elements to which the Empty value is assigned vary depending on how the array variable is declared.

Example 1:

If the number of elements is omitted in a one-dimensional array:

If you assign a value to the *m*-th element, the Empty value is assigned from the first element to the (*m*-1)-th element.

```
Dim A( )
A(1) = "abc"
```

(Element 1)

abc

```
A(5) = "def"#
```


 The elements from A(2) to A(4) are set to an Empty value.

(Element 1)	(Element 2)	(Element 3)	(Element 4)	(Element 5)
abc	Empty value	Empty value	Empty value	def

Example 2:

If a row element and a column element are omitted in a two-dimensional array:

When a value is assigned to the element in the *m*-th row of the *n*-th column, the Empty value is assigned to the elements in the first to (*m*-1)-th rows of the first column as well as to the elements in the first to (*n*-1)-th columns of the *m*-th row.

```
Dim A( , )
A(3,5) = "abc"#
```


 A(1,1), A(2,1), and the elements from A(3,1) to A(3,4) are set to an Empty value.

	(Column 1)	(Column 2)	(Column 3)	(Column 4)	(Column 5)
(Row 1)	Empty value				
(Row 2)	Empty value				
(Row 3)	Empty value	Empty value	Empty value	Empty value	abc

Blank elements, such as A(1,2), do not have a value stored in them. Any attempt to reference such an element results in an error with the following message: Cannot reference an unset index of a variable array variable.

Example 3:

If a column element is omitted in a two-dimensional array:

When a value is assigned to the element in the *m*-th row of the *n*-th column, the Empty value is assigned to the elements in the first to (*n*-1)-th columns of the *m*-th row.

```
Dim A(3, )
A(3, 5) = "abc" #
```

#

The elements from A(3, 1) to A(3, 4) are set to an Empty value.

	(Column 1)	(Column 2)	(Column 3)	(Column 4)	(Column 5)
(Row 1)					
(Row 2)					
(Row 3)	Empty value	Empty value	Empty value	Empty value	abc

Blank elements, such as A(1, 1), do not have a value stored in them. Any attempt to reference such an element results in an error with the following message: Cannot reference an unset index of a variable array variable.

Example 4:

If a row element is omitted in a two-dimensional array:

When a value is assigned to the element in the *m*-th row of the *n*-th column, the Empty value is assigned to the elements in the first to (*m*-1)-th rows of all columns as well as to the first to (*n*-1)-th columns of the *m*-th row.

```
Dim A(, 5)
A(3, 5) = "abc" #
```

#

The elements from A(1, 1) to A(1, 5), from A(2, 1) to A(2, 5), and from A(3, 1) to A(3, 4) are set to an Empty value.

	(Column 1)	(Column 2)	(Column 3)	(Column 4)	(Column 5)
(Row 1)	Empty value	Empty value	Empty value	Empty value	Empty value
(Row 2)	Empty value	Empty value	Empty value	Empty value	Empty value
(Row 3)	Empty value	Empty value	Empty value	Empty value	abc

(2) Data structure examples for array variables

Examples of data structures for array variables are shown below.

- In a one-dimensional array variable, the index numbers represent the elements.

Example:

An array variable declared as Dim A(5) has five elements as shown below.

(Element 1)	(Element 2)	(Element 3)	(Element 4)	(Element 5)
A(1)	A(2)	A(3)	A(4)	A(5)

- In a two-dimensional array variable, the first index number represents the row element and the second index number represents the column element.

Example 1:

An array variable declared as Dim B(1, 5) has one row with five columns as shown below.

	(Column 1)	(Column 2)	(Column 3)	(Column 4)	(Column 5)
(Row 1)	B(1, 1)	B(1, 2)	B(1, 3)	B(1, 4)	B(1, 5)

Example 2:

An array variable declared as Dim C(3, 4) has three rows with four columns each as shown below.

	(Column 1)	(Column 2)	(Column 3)	(Column 4)
(Row 1)	C(1,1)	C(1,2)	C(1,3)	C(1,4)
(Row 2)	C(2,1)	C(2,2)	C(2,3)	C(2,4)
(Row 3)	C(3,1)	C(3,2)	C(3,3)	C(3,4)

6.1.6 Constants

Table 6-3 summarizes the constants that have a special meaning in JP1/Script.

Table 6–3: Constants

Constant	Meaning
Empty	Indicates a null value ("").
True	Indicates a true value (non-zero).
False	Indicates a false value (zero).

6.1.7 Numeric coding conventions

The following coding conventions apply to numerics:

- 0, +0, and -0 are regarded as the same value.
- A sequence of all zeros are regarded as zero.
- JP1/Script handles integers only.
Commas (,) are not recognized when a numeric value is enclosed with double quotation marks ("). For example, "10,000" and 10000 are regarded as the same value.
- Plus signs preceding numerics are not recognized. For example, 600 and +600 are regarded as the same value.
- JP1/Script handles numerics in the range -2,147,483,647 to 2,147,483,647.
- A numeric value enclosed with double quotation marks (") is not handled as a string. For example, "10000" and 10000 are regarded as the same value. To interpret a numeric as a character string, use the `STR` command.

6.1.8 String coding conventions

The following coding conventions apply to strings:

- Strings containing characters other than numerics must be enclosed with double quotation marks (").
- A variable word or intrinsic constant enclosed with double quotation marks (") is regarded as a string. However, an intrinsic constant specified as an argument of a command is regarded as an intrinsic constant.
- JP1/Script handles strings of up to 1,024 bytes in length.
- Two double quotation marks ("") with nothing between them are interpreted as an Empty value.
- A quoted numeric that includes a string other than a comma (,) is interpreted as a string.

6.1.9 Operation conventions

The following conventions apply to operations in scripts:

- Operations on expressions enclosed in parentheses begin from the innermost expression.
- Variables are basically handled as string type. However, for an operation on strings consisting entirely of numerics, JP1/Script handles the strings as numerics and performs the operation.
- + and - are the only unary operators.

6.1.10 Operator precedence

Table 6-4 shows operator precedence.

Table 6–4: Operator precedence

Precedence	Operator	Description
High ↑	^	Exponential operator
	+, -	Unary arithmetic operators
	*, /, \, Mod	Multiplication, division, integer division, and remainder division (modulus) operators
	+, -	Addition and subtraction operators
	&	String concatenation
	=, <>, <, <=, >, >=	Comparison operators
	Not	Logical NOT
	And	Logical AND
	Or	Logical OR
↓ Low		

6.1.11 Script coding conventions

JP1/Script uses the following coding conventions:

- When part of a script (excluding a statement) spans two or more lines, write an underscore (`_`) as the final character. Put at least one space before the underscore. There is no need to write an underscore, however, when the command arguments are enclosed in parentheses.

Example: When the arguments are enclosed in parentheses:

```
Command (A, B, C,
        D, E, F)
```

Example: When the arguments are not enclosed in parentheses:

```
Command A, B, C, _
        D, E, F
```

- Command names must not extend over more than one line.

- Commands are case-insensitive.
- The maximum size of a script written as one line is 10,240 bytes. Data exceeding this size is ignored.
- The maximum size of the character strings written in a script is 1,024 bytes. Data exceeding this size is ignored.
- Write command arguments in one of the following forms. If writing more than one argument, delimit each argument with a comma or space.

Example: Argument specification (Δ : space):

```
Command (A)
Command  $\Delta$  (A)
Command  $\Delta$  A
Command (A, B)
Command  $\Delta$  (A, B)
Command  $\Delta$  A  $\Delta$  B
Command  $\Delta$  A, B
```

- When the argument of a command or function is itself a command or function that takes an argument, the arguments of that command or function written as the argument must be enclosed in parentheses.

Example: Argument specification when the argument must be enclosed in parentheses:

```
Command (Function (A, B, C) , D, E)
Command Function (A, B, C) , D, E
Command Function (A, B, C) D  $\Delta$  E
```

- When the right or left side of an operator is a command or function that takes an argument, the arguments must be enclosed in parentheses.

Example: `M=Command(A, B, C) + Function(D)`

When specifying the version of Script Engine for executing a script, the following must be written at the head of each script file, where *VV* is the JP1/Script version and *RR* is the JP1/Script revision:

```
#FileVersion=VVRR
```

If the version specification is omitted or if a nonexistent version is specified, the script file is executed with the version that was set in the Manager window's Options (Compatibility) dialog box. If this line is omitted, the script file will be executed by the version of the Script Engine set in the Manager window's Options (Compatibility) dialog box. When you create a new script file, you can specify in the Manager's Create New Script File dialog box whether to automatically insert the Script Engine version at the head of the file.

Example: `#FileVersion = 0700`

The Script Engine version does not need to be the same as the version of the installed JP1/Script program.

You can apply the conventions of the specified version of Script Engine to execute a script file.

For example, if JP1/Script 06-00 that uses `CallSpt` as the `Function` procedure name is upgraded to 06-71 or a later version, you can execute a script file without any changes if the value of `#FileVersion` is still 0600. If you change the value of `#FileVersion` to 0671, the execution result will be invalid.

- The branch destination written for a `GoTo` statement, `On Error` statement, or other control must end with a colon (:).

Example: `LabelName :`

- To avoid converting `\r`, `\n`, `\t`, or `\\` into the corresponding control code when included in a character string, you must write the following at the head of the script file:

```
#Option=NOCHANGE
```

The above line must follow the line `#FileVersion=VVRR`, if included.

If #Option=NOCHANGE is missing, \r, \n, \t, and \\ will be converted into the corresponding control code. That is, \r will be converted into a CR (carriage return) code, \n into a CR-LF (carriage return and line feed) code, \t into a tab, and \\ into a single backslash (\).

Note that version 0500 and earlier versions of Script Engine change \r, \n, \t, and \\ into the corresponding control codes unconditionally, even if #Option=NOCHANGE is written at the head of the file.

Example 1:

```
#FileVersion = 0700
#Option = NOCHANGE
MessageBox("C:\\WkDir\\Script.SPT")
```

The above example displays the following string in a message box:

```
C:\\WkDir\\Script.SPT
```

Example 2:

```
#FileVersion = 0700
MessageBox("C:\\WkDir\\Script.SPT")
```

The above example displays the following string in a message box:

```
C:\\WkDir\\Script.SPT
```

- In the following commands, an ampersand (&) included in the passed string is not displayed. Instead, the character after the ampersand is underscored.
 - Message command for displaying a message in a window
 - InputBox command for displaying a message and text box title in a dialog box

To display an included ampersand as is, write #Option=NOPREFIX at the head of the script file. If you have already written #Option=NOCHANGE, add the NOPREFIX keyword, delimiting the keywords with a comma as follows:

```
#Option=NOCHANGE,NOPREFIX
```

You can specify the two keywords in any order. Write this line immediately below the #FileVersion=VVR line, if included.

Note that if the Script Engine version is earlier than 0520, a single ampersand will be displayed as an underscore added to the next character, even if you include #Option=NOPREFIX. To display an & character as is, use two ampersands (&&).

Example 1:

```
#FileVersion = 0700
#Option = NOCHANGE,NOPREFIX
Message(Target_Dispon,"Display","Client&Server")
```

The above code displays the following text in a window:

```
Client&Server
```

Example 2:

```
#FileVersion = 0700
Message(Target_Dispon,"Display","Client&Server")
```

The above code displays the following text in a window:

```
ClientServer
```

- To display n double quotation marks (") as is, write $n*2$ double quotation marks in the string passed to the command.

Example:

```
MessageBox("Error code:""99"")
```

The above code displays the following text in a message box:

```
Error code:"99"
```

6.1.12 JP1/Script exit codes

At script execution, the value set in the `Exit` or `ExitWindows` command is usually returned as the exit code. If you did not use these commands or omitted the command argument, 0 is returned as the exit code. However, if script execution results in an error, a JP1/Script exit code may be returned.

Table 6-5 shows the values of exit codes set in the following registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\ExitCode
```

Table 6–5: Exit codes set in the registry

Value name	Default	Meaning
<code>AlreadyRun</code>	16	The specified script file has already started.
<code>Terminate</code>	17	The JP1/Script process was forcibly terminated.
<code>TimeOut</code>	18	The JP1/Script process was terminated because the timeout interval elapsed.
<code>GrammarError</code>	19	A syntax error occurred.
<code>ExAbortError</code>	20	An error canceled the JP1/Script process. (Possible errors include memory shortage, reference to an undefined variable, nonexistent procedure, and statement error.)
<code>Error</code>	99	An error occurred before the JP1/Script process started. (For example, the specified script file was not found.)
--	21 to 32	Reserved for future use.

Important note

Be careful not to specify a JP1/Script exit code in the `Exit` command or `ExitWindows` command.

For example, if you specify exit code 19 (the default of the `GrammarError` value in the registry in Table 6-5) in these commands, the following message will be output to the event log even if the script execution terminates normally: `Script execution has finished. Syntax error occurred. (script-file-name)`.

6.1.13 JP1/Script event logs

JP1/Script outputs event logs according to the status. The user can also use the `MessageEventLog` command to specify output of event logs.

Table 6-6 lists and describes the event IDs and event logs that are output. The information output source is JP1/Script.

Table 6–6: Event IDs and logs

Event ID	Type	Event log	Output information#1
1	Information	Information log at the beginning of script execution	Script execution has started. (<i>script-file-name</i>)
2	Information	Information log at the end of script execution	Script execution has finished. (<i>script-file-name</i>)
3	#2	User-specified log using the <code>MessageEventLog</code> command	Message specified in the <code>MessageEventLog</code> command
4	Information	Information log for the execution of Script Launcher	The Script launcher is not running on the Remote Desktop service client.
	Warning	Warning log for the execution of the JP1/Script service, Script Launcher service, or Script Launcher	Warning message (error message)
	Error	Error log for the execution of the JP1/Script service, Script Launcher service, or Script Launcher	JP1/Script may be installed incorrectly. Reinstall JP1/Script.
			The contents of the automatic start information file are invalid. Remake the automatic start information file. (<i>automatic-start-information-file-name</i>)
			The Script Launcher service is already running. Script Launcher cannot be started.
			Script Launcher is already running. The Script Launcher service cannot be started.
			The startup account for the Script Launcher service is not a local system account. The Script Launcher service cannot be started.
			A logon user name is not specified in the startup parameter for the Script Launcher service. The Script Launcher service cannot be started.
The NetExec thread for the Script Launcher service cannot be started because the content of the policy file limited to the NetExec command is invalid. Re-create the policy file limited to the NetExec command and restart the Script Launcher service. (<i>file-name</i>)			
An error message for the error that occurred.			
5	Warning	Warning log at execution of the <code>ExitWindows</code> command	The <code>ExitWindows</code> (<i>type-of-shutdown-operation-specified-in-the-command</i>) command was executed. (<i>script-file-name</i>)
			Executing the <code>ExitWindows</code> command failed. (<i>function-name, error code</i>)

Event ID	Type	Event log	Output information#1
6	Warning	Warning log for unsuccessful issuance of an event to JP1/IM or JP1/Base	An error occurred in a JP1/Integrated Manager function. (<i>JP1/IM-or-JP1/Base-function-name:: JP1/IM-or-JP1/Base-function-error-condition-code</i>)
7	Information	Information log at execution of the TerminateProcess command	The TerminateProcess (Id= <i>process-ID-specified-in-the-command</i>) command was executed. (<i>script-file-name</i>)
8#3	Information	Information log at the beginning and end of the JP1/Script service, Script Launcher service, and Script Launcher execution	<p>The Script Launcher will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The Script Launcher will now end. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread of the Script Launcher will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread of the Script Launcher will now end. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The Script Terminate Control Launcher was executed. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The JP1/Script service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The JP1/Script service will now end. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread of the JP1/Script service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread (which limits users who can execute the NetExec command) for the JP1/Script service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread of the JP1/Script service will now end. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The File Mapping Object Control thread of the JP1/Script service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The File Mapping Object Control thread of the JP1/Script service will now end. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The Script Launcher service will now start. (<i>date time elapsed-time-since-system-startup</i>) (<i>execution-account-name</i>)</p> <p>The Script Launcher service will now end. (<i>date time elapsed-time-since-system-startup</i>) (<i>execution-account-name</i>)#5</p> <p>The NetExec thread for the Script Launcher service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p> <p>The NetExec thread (which limits users who can execute the NetExec command) for the Script Launcher service will now start. (<i>date time elapsed-time-since-system-startup</i>)</p>

Event ID	Type	Event log	Output information#1
8#3	Information	Information log at the beginning and end of the JP1/Script service, Script Launcher service, and Script Launcher execution	The NetExec thread for the Script Launcher service will now end. (<i>date time elapsed-time-since-system-startup</i>)
16	Error	Error log at startup of multiple scripts	The specified script file has already been started. (<i>script-file-name</i>)
17	Warning	Warning log at forced termination of a script	Forcibly terminated the JP1/Script process. (<i>script-file-name</i>)
18	Warning	Warning log at script termination due to timeout	Terminated the JP1/Script process because a timeout occurred. (<i>script-file-name</i>)
19	Information	Information log at script termination due to a syntax error	Script execution has finished. Syntax error occurred. (<i>script-file-name</i>)
96	Warning	Warning log when the JP1/Script service is not running at startup of script execution	The execution result will not be output to the analysis/execution trace file because the JP1/Script service is not running. \nAlso, execution of the Message, EntryStartUp, or CancelStartUp command, or the global variable operation command will result in an error. \nStart the JP1/Script service, and then perform execution. (<i>script-file-name</i>)
98#4	Error	Error log when no analysis trace or execution trace (/SPALV(0) or /SPXLV(0) specified) is output	User name: <i>user-name-at-script-execution</i> Script file name: <i>script-file-name</i> The following error occurred. <i>Line-number-of-the-error</i> line; <i>error-message-for-the-error-that-occurred</i>
99	Error	Error log at the beginning of script execution	An error message for the error that occurred.

#1

Italic letters indicate a placeholder whose contents depend on the log. *date*, *time*, and *elapsed-time-since-system-startup* are displayed in the formats *yyyy/mm/dd*, *hh:mm:ss.ss*, and in milliseconds, respectively.

#2

The event type specified by the `MessageEventLog` command.

#3

To suppress output of this event log, set the registry key as follows:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1\Script\SPTF\SPV

Value name

CommandLine (*character-string*)

Value datatype

REG_SZ

Value

/NOEVLOG (8)

When the setting takes effect

The setting takes effect the next time the JP1/Script service, the Script launcher, or the Script launcher service is started.

#4

To output this error to the event log, set the registry key as follows:

Registry key

HKEY_CURRENT_USER\SOFTWARE\HITACHI\JP1\Script\SPTX

Value name

ErrorEventLog

Value datatype

REG_DWORD

Value

1: The error is output to the event log.

Other than 1: No error is output to the event log.

When the setting takes effect

The setting takes effect the next time the script file is executed.

#5

When an error occurs on startup of the Script Launcher service, an event log entry is output without an execution account name.

Of these event logs, you can suppress output of non-error event logs. To specify output suppression, see *If you choose Command Line:* in 6.2.2(4) */NOEVLOG (or /noevlog)*.

6.1.14 JP1 Events issued by JP1/Script

Table 6-7 lists and describes the JP1 events that are issued by JP1/Script.

Table 6–7: JP1 events issued by JP1/Script

Event ID	Severity	Description of event	Event occurrence	Output information (see Table 6-8)
00003B00#	Information	Script start/end event	When script execution begins and ends	C1 to C5, C8, C9
00003B01	Error	Memory access error	When a memory access error occurs	C1 to C5, C8
00003B03	User-specified value	Event generated by IMEventMessage command	When IMEventMessage command is executed	C1 to C9
00003B09#	Information	Command normal termination event	When command terminates normally	C1 to C5, C8, C9
00003B10#	Error	Command error termination event	When command terminates with an analysis error	C1 to C5, C8, C9
00003B11#	Error	Command error termination event	When command terminates with an execution error	C1 to C5, C8, C9

#

This event is issued when display is enabled on the **JP1/IM** page in the Options dialog box, which is displayed by going to the manager's **Tools** menu and choosing **Options**.

Table 6–8: Output information

Symbol	Output information
C1	Severity
C2	User name
C3	Product name
C4	Registration type name
C5	Registration name

Symbol	Output information
C6	Event type
C7	Start time
C8	End time
C9	Termination code

6.2 Rules for writing command lines

This section explains how to write a command line for executing a script file.

You must specify a command line in the Set Execution Environment (Start Information) dialog box if you want to:

- Specify parameters for executing the script file
- Specify whether to output an analysis trace file (output by default)
- Specify whether to output an execution trace file (output by default) and the output level
- Suppress output of an event log (output by default)

6.2.1 Command line formats

Command lines take one of the following three forms.

(1) Command line written in the Set Execution Environment dialog box

```
[parameter . . .] [/SPALV (n)] [/SPXLV (n)] [/NOEVLOG [ (n [, n] . . . ) ]]
```

The script file is already selected, so there is no need to specify its name in the command line.

Parameters specified in a command line are stored in the location variables.

For details about location variables, see *G. Glossary*.

(2) Script file called by specifying parameters in the Exec or NetExec command

```
Exec (script-file-name, whether-to-wait-for-termination [, "parameter" . . .] [, "/SPALV (n) " ] [, "/SPXLV (n) " ] [, "/NOEVLOG [ (n [, n] . . . ) " ]])
```

Parameters specified in a command line are stored in the location variables.

For details about position variables, see *G. Glossary*.

(3) Command line written in executable form (SPTXE.EXE) in a user program

```
SPTXE.EXE script-file-name [parameter . . .] [/SPALV (n)] [/SPXLV (n)] [/NOEVLOG [ (n [, n] . . . ) ]]
```

Parameters specified in a command line are stored in the location variables.

For details about position variables, see *G. Glossary*.

6.2.2 Command line parameters

(1) Parameters

Parameters specified in command lines are stored in location variables `%n` (where *n* is a positive integer) that are handled by script files. `%0` is the script file name (full path), and `%1` is the first position parameter (where `%n` is the *n*th parameter).

If specifying multiple parameters, follow the rules for writing command lines. For details, see [6.2.3 Command line coding conventions](#).

The folder name `Program Files` used in the following examples includes a blank.

Example 1:

Command line written in the Set Execution Environment dialog box

Command line format in the dialog box:

```
ABC 123 "C:\Program Files\"
```

Values assigned to the location variables:

```
%0:script-file-name
%1:ABC
%2:123
%3:C:\Program Files\
```

Example 2:

Script file called by specifying parameters in the Exec or NetExec command

Command line format:

```
Exec ("C:\Temp\Test.SPT", True, "ABC", "123", ""C:\Program Files\ "")
```

Values assigned to the location variables:

```
%0:C:\Temp\Test.SPT
%1:ABC
%2:123
%3:C:\Program Files\
```

Example 3:

Command line written in executable form (`SPTXE.EXE`) in a user program

Specification in the user program:

```
SPTXE.EXE C:\Temp\Test.SPT ABC 123 "C:\Program Files\"
```

Values assigned to the location variables:

```
%0:C:\Temp\Test.SPT
%1:ABC
%2:123
%3:C:\Program Files\
```

(2) /SPALV(n) (or /spalv(n))

Use this parameter to specify whether to output an analysis trace file. A trace file is output by default when this parameter is omitted. For n , specify 0 or a greater integer. The default is a nonzero value.

- $n = 0$: Does not output an analysis trace file.
- $n = \text{Non-zero value}$: Outputs an analysis trace file.

Example:

Specify as follows to suppress output of an analysis trace file.

```
SPTXE.EXE C:\Temp\Test.SPT ABC 123 "C:\Program Files\" /SPALV(0)
```

(3) /SPXLV(n) (or /spxlv(n))

Use this parameter to indicate whether to output an execution trace file and to specify the output level. When this parameter is omitted, only error results are output to the trace file. For n , specify an integer from 0 to 3. The default is 1. If you specify 2 or 3, execution performance of script files will be degraded because the amount of output information will increase.

- $n = 0$: Does not output an execution trace file.
- $n = 1$: Outputs only error results to the execution trace file.
- $n = 2$: Outputs error and normal results to the execution trace file.
- $n = 3$: Outputs error and normal results, and command start and end times, to the execution trace file.

Example:

Specify as follows to output the command start and end times to the execution trace file.

```
SPTXE.EXE C:\Temp\Test.SPT ABC 123 "C:\Program Files\" /SPXLV(3)
```

(4) /NOEVLOG (or /noevlog)

Use this parameter to suppress event log output. If you omit this parameter, all event logs are output. For details about the types of event logs, see [6.1.13 JP1/Script event logs](#).

If this parameter is omitted, all event logs are output.

/NOEVLOG (or /noevlog)

Outputs an event log for error events only.

/NOEVLOG (n, n, \dots) (or /noevlog (n, n, \dots))

Disables log output for the event IDs specified in n . Event IDs can be specified in any order. Enter commas to delimit the IDs.

Example:

Specify as follows to suppress log output for event ID 3 (user-specified log using the MessageEventLog command).

```
SPTXE.EXE C:\Temp\Test.SPT ABC 123 "C:\Program Files\" /NOEVLOG(3)
```

6.2.3 Command line coding conventions

The following rules apply to writing command lines:

- Delimit multiple parameters with a space.
- To specify a parameter containing a space that is not a delimiter, enclose the entire parameter with double quotation marks ("").
- A maximum of 245 parameters can be set in a command line. Parameters exceeding this maximum are ignored.
- A maximum of 1,024 one-byte characters can be specified for a parameter. Characters exceeding this maximum are ignored.

The first two of these rules also apply when calling an executable file other than a script file from the `Exec` or `NetExec` command.

6.2.4 Notes on writing command lines

The following notes apply to writing command lines:

- The values specified in a command line apply only to the specific script file. To apply command line values to all the script files on the computer, set the values in the following registry:

Registry key

HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\SPTF\SPV

Value name

CommandLine

Value datatype

REG_SZ

Value

Value in a command line

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note

When a command line is specified in the dialog box and in this registry, both specifications are valid. However, if different values are specified in a parameter, such as `/SPXLV(3)` in the dialog box and `/SPXLV(0)` in the registry, the specification in the dialog box takes precedence.

- Do not suppress event log output for the JP1/Script service, Script Launcher, and Script Launcher service by setting a command line in the execution environment file. Instead, specify `/NOEVLOG` or `/NOEVLOG(4)` in the above registry.
- Unlike `/SPALV(n)` and `/SPXLV(n)`, `/NOEVLOG` and `/NOEVLOG(n,n,...)` specified in a command line in the Set Execution Environment (Start Information) dialog box or Set Execution Environment (Command Line) dialog box are interpreted as location variables. If you do not want them interpreted as location variables, set 1 for the registry key shown below.

Irrespective of the registry value, `/NOEVLOG` and `/NOEVLOG(n,n,...)` are not interpreted as location variables if they are specified in a command line written in executable form (`SPTXE.EXE`) in a user program used to call a script. This also applies if they are specified for `CommandLine` in the registry.

Registry key

```
HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\SPTX
```

Value name

```
NoEvLog_Opt
```

Value datatype

```
REG_DWORD
```

Value

0: Interpret /NOEVLOG or /NOEVLOG (*n, n, . . .*) as a location variable (initial value).

1: Do not interpret /NOEVLOG or /NOEVLOG (*n, n, . . .*) as a location variable.

When the setting takes effect

The setting takes effect the next time the script file is executed.

- To include *n* double quotation marks (") as is in a parameter, write *n*2* double quotation marks and enclose the entire parameter with double quotation marks.[#]

At this time, set 1 for the registry key shown below.[#]

To use the NetExec command, also set 1 in the registry on the computer you are calling.

Registry key

```
HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\SPTX
```

Value name

```
DQString_Opt
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") and enclose the entire parameter with double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") and enclose the entire parameter with double quotation marks (") is enabled.

When the setting takes effect

The setting takes effect the next time the script file is executed.

If a value other than 1 is specified for the above registry key, the number of double quotation marks (") that must be specified varies depending on the execution format of the script file.

- To use a command line executable (SPTXE.EXE) in a user program to launch a script file:
Specify *n*4* double quotation marks (") and two double quotation marks (") enclosing the entire parameter.
Example: Specify a"b for the parameter string.

```
SPTXE.EXE SPT.spt "a""""b"
```
- To specify the parameter of the Exec command to call a script:
Specify *n*4*2+2* double quotation marks (") and two double quotation marks (") enclosing the entire parameter.
Example: Specify a"b for the parameter string.

```
Exec (_SCF_+"ABC.SPT", True , "a""""""""""b")
```
- To specify the parameter of the NetExec command to call a script file:
Specify *n*4*2*2*2* double quotation marks ("), and 15 double quotation marks (") before and after the parameter.

Example: Specify a"b for the parameter string.

```
NetExec ( compl,_TEMP_"NETWORK.SPT" ,True , ,False , _  
, "a"b")
```

#

If you do not specify this registry, the specification requirements become complicated. It is recommended that you specify this registry when you are developing a new system.

- If you set parameters in a command line used for starting a script and in a command line in the script execution environment file, both specifications take effect. In this case, the parameters specified in the command line used for starting the script are applied first.

7

Statements

This chapter describes the statements you can use when creating a script.

7.1 List of statements

Table 7-1 lists the statements that you can use when creating a script. The numbers in parentheses in the statement list indicate the product version from which the statement is supported. Statements without a following number are supported from version 01-00.

Table 7–1: List of statements

Statement	Description
=	Assigns the value on the right side to the variable on the left.
Do...Loop (05-10)	Iterates a sequence of statements as long as a specified condition is true, or until a specified condition becomes true.
For...Next	Iterates a sequence of statements a specified number of times.
For...End For	Iterates a sequence of statements for all the files in a specified folder.
If...Then...Else	Executes conditional processing dependent on the value of an expression.
Select Case	Executes one of a number of statement blocks, depending on the value of a conditional expression.
While...End	Iterates a sequence of statements as long as a specified condition is true.
Function	Declares the name and arguments, and indicates the start of the Function procedure. It also defines a sequence of statements enclosed by the Function statement and End Function statement as a Function procedure.
Sub	Declares the name and arguments, and indicates the start of the Sub procedure. It also defines a sequence of statements enclosed by the Sub statement and End Sub statement as a Sub procedure.
Call	Passes control to a Sub procedure or Function procedure.
Exit <i>xx</i>	Exits from a While...End loop, For...Next loop, For...End For loop, Do...Loop loop, Function procedure, or Sub procedure.
GoTo (05-00)	Jumps to a specified label.
Continue (05-00)	Returns to the start of a While...End loop, For...Next loop, For...End For loop, or Do...Loop loop.
On Error (05-00)	Enables or disables transfer of control to a specified label if an error occurs.

7.1.1 =

Purpose

Assigns the value on the right side to the variable on the left.

Syntax

```
Result = Expression
```

Arguments

Result

Write the variable name in which to set the value.

Expression

Write any expression.

Description

The equals operator (=) assigns the value of *Expression* to the variable *Result*.

If the value of *Expression* is Empty, an Empty value is assigned to the variable *Result*. If the value is assigned to a variable that has not been declared with the Dim command, the system automatically allocates the variable and then assigns the value.

Examples

```
' Store "ABCDE" in variable result1.  
result1 = "ABCDE"  
  
' Store 200 in variable result2.  
result2 = 50 + 150  
  
' Store the current date in variable result3.  
result3 = Date
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.2 Do...Loop

Purpose

Flow control statement that iterates a sequence of statements as long as a specified condition is true, or until a specified condition becomes true.

Syntax

```
Do  
  [Statements]  
  [Exit Do]  
  [Statements]  
Loop [{While | Until} Condition]
```

Arguments

Statements

Write one or more statements to be iterated while *Condition* is true or until it becomes true. To write more than one statement, put a line feed after each statement.

Condition

Write a conditional expression that evaluates to true or false.

Description

The Do . . . Loop statement iterates a sequence of statements as long as *Condition* is true with the keyword While, or until *Condition* becomes true with the keyword Until.

The Exit Do statement can only be used within a Do . . . Loop control structure as a means of exiting the Do . . . Loop under a condition other than specified in *Condition*. You can write any number of Exit Do statements anywhere in the Do . . . Loop. Exit Do is often used with the evaluation of some condition (If . . . Then statement, for example) to pass control to the statement immediately following the Loop statement.

You can nest Do . . . Loop statements by placing one Do . . . Loop within another. Execution of an Exit Do in a nested structure enables escape from the innermost loop enclosing the Exit Do.

Unlike the `While...End` statement, a sequence of `Do...Loop` statements is executed at least once in a post-test structure. If you want to evaluate a condition before all the statements are executed, use a `While...End` statement.

Example

```
' Reverse returns the value of the inverted string.
' "EDCBA" is stored in result.
result = Reverse("ABCDE")
MessageBox(result)

Function Reverse(chrValue)
    Dim chrString, chrLength
    cnt = 0
    chrLength = Len(chrValue)
    Do
        chrString = chrString + Mid(chrValue, chrLength - cnt, 1)
        cnt = cnt + 1
    Loop While(cnt < chrLength)
    Reverse = chrString
End Function
```

JP1/Script version

Supported from JP1/Script 05-10.

7.1.3 For...Next

Purpose

Flow control statement that iterates a sequence of statements a specified number of times.

Syntax

```
For Counter = Start To End [Step Step]
    [Statements]
    [Exit For]
    [Statements]
Next
```

Arguments

Counter

Specify the variable storing the numeric value that is used as the loop counter. Comply with the variable naming conventions. You cannot specify an array variable or an element of an array variable for this variable.

Start

Specify the initial value of *Counter*.

End

Specify the final value of *Counter*.

Step

Specify the value by which *Counter* is incremented at each loop iteration. If you omit the `Step` argument, 1 will be added to *Counter* at each loop iteration. You can specify a positive or negative number in *Counter*. Loop execution is controlled as follows, according to the value you specify in `Step`.

Value	Execution condition
Positive or zero	$Counter \leq End$
Negative	$Counter \geq End$

Statements

Specify a group of statements to execute in the loop. Write the statements between `For` and `Next`. These statements will be executed the number of times specified in `For . . . Next`. You can write multiple statements in *Statements*, with a line return after each one.

Description

When program execution moves to the loop and all the statements in the loop have been executed, the *Step* value is added to *Counter*. At this time, if the loop execution condition is satisfied, the statements in the loop are executed again. If the loop execution condition is not satisfied, the program exits the loop and control passes to the statement following the `Next` statement.

The `Exit For` statement can only be used within a `For . . . Next` or `For . . . End For` control structure as a means of exiting a `For . . . Next` loop under a condition other than the specified iteration count. You can write any number of `Exit For` statements anywhere in the `For . . . Next` loop. `Exit For` is often used with the evaluation of some condition (`If . . . Then` statement, for example) to pass control to the statement immediately following `Next`.

You can nest `For . . . Next` statements by placing one `For . . . Next` within another. When nesting `For . . . Next` loops, specify a different variable name in each *Counter*.

Example

```

For I = 1 To 10
  For J = 1 To 10
    For K = 1 To 10
      . . .
    Next
  Next
Next

```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.4 For...End For

Purpose

Flow control statement that iterates a sequence of statements for all the files in a specified folder.

Syntax

```

For VarName = Filemask Do
  [Statements]
  [Exit For]
  [Statements]
End [For]

```

Arguments

VarName

Specify a variable name for storing the file names. Comply with the variable naming conventions. You cannot specify an array variable or an element of an array variable.

Filemask

Specify the file names you want to search by full path. A wildcard can be included. Enclose the full path with double quotation marks (").

Statements

Write the sequence of statements to be executed in the loop. Write the statements between `For` and `End For`. These statements are executed repeatedly until no more files can be found in the path specified in the `For...End For`.

You can write multiple statements in *Statements*, with a line feed after each one.

Description

The `For...End For` statement iterates a sequence of statements for all the files in a specified folder. When no more files can be found in the specified path, control moves to the statement following the `End For` statement.

The `Exit For` statement can only be used within a `For...End For` or `For...Next` control structure as a means of immediately exiting a `For...End For` loop. You can write any number of `Exit For` statements anywhere in the `For...End For` loop.

You can nest `For...End For` statements by placing one `For...End For` within another. When nesting `For...End For` loops, specify a different variable name in each *VarName*.

When `*.*` is specified in *Filemask*, file names other than `.` and `..` are stored in *VarName*.

Supplementary note

- When the system searches for the files in a `For...End For` loop, it also retrieves any short file names converted into MS-DOS file format and saved for compatibility.
- For the `For...End For` loop, the file search order is not guaranteed.

Example

```
' Back up the script files in the execution folder to
' another folder.
Dim path1 ,bkupDir
bkupDir = _BIN_"BKUP\"
MakeDir (bkupDir)

' Search for script files.
For path1 = _BIN_"*.*SPT" Do
  ' Ignore folder.
  If IsExistDir(_BIN_+path1) = False Then
    ' Back up to another folder.
    Copy(_BIN_+path1 ,bkupDir+path1)
  End If
End For
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.5 If...Then...Else

Purpose

Flow control statement that executes conditional processing dependent on the value of an expression.

Syntax 1

```
If Condition Then
    [Statements]
[Else
    [ElseStatements]]
End [If]
```

Syntax 2

```
If Condition Then
    [Statements]
[ElseIf Condition-n Then
    [ElseifStatements]]...
[Else
    [ElseStatements]]
End [If]
```

Arguments

Condition

Conditional expression that evaluates to true or false.

Statements

Statements to be executed if *Condition* is true. You can write multiple statements in *Statements*, with a line return after each one.

Condition-n

Same as the *Condition* argument.

ElseifStatements

Specify a group of statements to be executed if *Condition-n* is true. You can write multiple statements in *ElseifStatements*, with a line return after each one.

ElseStatements

Specify a group of statements to be executed if none of the conditions (*Condition* or *Condition-n*) defined before *Else* is true. You can write multiple statements in *ElseStatements*, with a line return after each one.

Description

The *Condition* in the *If...Then...Else* statement is evaluated first. If *Condition* is true, the statements following *Then* are executed. If *Condition* is false, the *Else* clause (if any) is executed in the first syntax. In the second syntax, the conditions (*Condition-n*) specified in the *ElseIf* clauses are evaluated. If any of these conditions is true, the statements following the associated *Then* are executed. If all of the *ElseIf* conditional expressions are false (or if there are no *ElseIf* clauses), the statements following *Else* are executed. After executing the statements following *Then* or *Else*, program execution continues from the statement following *End*.

You can define *Else* and *ElseIf* clauses as required. In the second syntax, you can specify as many *ElseIf* clauses as you wish in the *If* block, but no *ElseIf* clauses are allowed after the *Else* clause. You can also place *If* statements in a nested structure.

JP1/Script version

Supported from JP1/Script 01-00.

7.1.6 Select Case

Purpose

Flow control statement that executes one of a number of statement blocks, depending on the value of a conditional expression.

Syntax

```
Select Case TestExpression
[Case ExpressionList-n
  [Statements-n]]...
[Case Else
  [ElseStatements-n]]
End [Select]
```

Arguments

TestExpression

Write any conditional expression.

ExpressionList-n

Mandatory if you write a Case clause. Specify one or more expressions, delimited with commas.

Statements-n

Statements to be executed if *TestExpression* matches any one of *ExpressionList-n*. You can write multiple statements in *Statements-n*, with a line return after each one.

ElseStatements

Statements to be executed if *TestExpression* does not match any of the Case clauses. You can write multiple statements in *ElseStatements*, with a line return after each one.

Description

If *TestExpression* matches any *ExpressionList* in a Case clause, the statements following that Case clause are executed up to the next Case clause or up to the End Select statement. When the block has executed, control passes to the statement following End Select. If *TestExpression* matches more than one Case clause, only the statements following the first match are executed.

The Case Else clause specifies the *ElseStatements* to be executed if *TestExpression* does not match any *ExpressionList* in any of the Case clauses. Although a Case Else clause is not always required, it is a good idea to write a Case Else statement in a Select Case block to process unexpected *TestExpression* values. If no *ExpressionList* in any of the Case clause matches *TestExpression* and there is no Case Else statement, execution continues from the statement following End Select.

Select Case statements can be nested. Each nested Select Case statement must have a corresponding End Select statement.

Example

```
' Determine argument Result, create and output a message to "Result.txt".
Sub LogOutput ( Result )
  Dim MessageText
  Select Case Result
  Case "OK"
    MessageText = " End normally. " & "Status=[" & Result & "]"
  Case "NG"
    MessageText = " End abnormally. " & "Status=[" & Result & "]"
  Case Else
    MessageText = " An unexpected error occurred. " & _
                  "Status=[" & Result & "]"
```

```

End Select
Message ( Target_File , _SCF_+"Result.txt" ,MessageText _
        ,128 ,1024 )
End Sub

```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.7 While...End

Purpose

Flow control statement that iterates a sequence of statements as long as a specified condition is true.

Syntax

```

While Condition
  [Statements]
  [Exit While]
End [While]

```

Arguments

Condition

Write a conditional expression that evaluates to true or false.

Statements

Write one or more statements to be executed while *Condition* is true. Begin each statement on a new line.

Description

If *Condition* is true, all the statements in *Statements* are executed until the End statement is reached. Control then returns to the While statement and *Condition* is again evaluated. If *Condition* is still true, the process is repeated. If *Condition* is not true, execution moves to the statement following the End statement.

The Exit While statement can only be used within a While...End control structure as a means of exiting the While...End loop under a condition other than specified in *Condition*. You can write any number of Exit While statements anywhere in the While...End loop.

While...End loops can be nested to any level. Each End statement corresponds to the most recently executed While statement.

Example

```

' Search for specified data among all the elements of a
' two-dimensional array variable.
Dim array1( , )
Dim line
:
(Store the values in array variable array1)
:
line = 0
allCnt = GetArrayCount ( array1 )
While line < allCnt
  buff = InArray ( array1( line + 1 ) , "1999" , 1 , False )
  If 0 < buff Then
    Exit While
  End If

```

```
    line = line + 1
End while
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.8 Function

Purpose

Declares the name and arguments, and indicates the start of the `Function` procedure. It also defines a sequence of statements enclosed by the `Function` statement and `End Function` statement as a `Function` procedure.

Syntax

```
Function Name [(ArgList)]
  [Statements]
  [Name = Expression]
  [Exit Function]
  [Statements]
  [Name = Expression]
End [Function]
```

Arguments

Name

Write the name of the `Function` procedure you are defining. Comply with the variable naming conventions.

ArgList

Specify a list of variables representing the arguments to pass to the `Function` procedure when it is called. Delimit multiple variables with commas. Enclose any arguments in parentheses.

Statements

Write a group of statements to be executed within the `Function` procedure.

Expression

Write the return value of the `Function` procedure.

Description

`Function` procedures are public and can be referenced by all other procedures in the script.

The executable code must be written entirely within the procedure. You cannot define a `Function` procedure inside another `Function` procedure or inside a `Sub` procedure.

The `Exit Function` statement causes the program to immediately exit from a `Function` procedure. Execution continues from the statement following the statement that called the `Function` procedure. You can write any number of `Exit Function` statements anywhere in a `Function` procedure.

For details on calling `Function` procedures, see *7.1.10 Call*.

To return a value from a `Function` procedure, assign the value to the *Name* of the `Function` procedure. You can assign any number of values to *Name* anywhere within the procedure. If no value is assigned to *Name*, the procedure returns a zero-length string ("") as a set return value.

Two types of variables can be used in `Function` procedures: variables that are explicitly declared within the procedure and those that are not. Variables that are explicitly declared in a procedure, using the `Dim` command or equivalent, behave as local variables that are valid only within that procedure. Variables that are used but not explicitly declared in a procedure are also local unless they are explicitly declared at some higher level outside the

procedure. The value of local variables in a `Function` procedure are discarded when the procedure finishes execution.

A variable that is not explicitly declared in the procedure can be used in a `Function` procedure, but it will be handled as a global variable if another variable or entity is defined at the script level with the same name as the undeclared variable.

Example

```
' Compare the input data and display the results in ascending
' order.
Dim compnum1,compnum2
InputBox ( "Enter the numeric values to be compared.", "Input data", _
          100, 100, compnum1, "numeric value 1" ,compnum2, "numeric value
2" )

If NumberCompare( compnum1 , compnum2) <> True Then
    lower = compnum2
    upper = compnum1
Else
    lower = compnum1
    upper = compnum2
End If
MessageBox( "lower=[" & lower & "]" , upper=[" & upper & "]" )

Function NumberCompare(p1, p2)
    NumberCompare =True
    Dim lower,upper
    lower = p1
    upper = p2
    If lower > upper Then
        NumberCompare = False
        Exit Function
    End If
End Function
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.9 Sub

Purpose

Declares the name and arguments, and indicates the start of the `Sub` procedure. It also defines a sequence of statements enclosed by the `Sub` statement and `End Sub` statement as a `Sub` procedure.

Syntax

```
Sub Name [(ArgList)]
    [Statements]
    [Exit Sub]
    [Statements]
End [Sub]
```

Arguments

Name

Write the name of the Sub procedure you are defining. Comply with the variable naming conventions.

ArgList

Specify a list of variables representing the arguments to pass to the Sub procedure when it is called. Delimit multiple variables with commas. Enclose any arguments with parentheses.

Statements

Write a group of statements to be executed within the Sub procedure.

Description

Sub procedures are public and can be referenced by all other procedures in the script.

The executable code must be written entirely within the procedure. You cannot define a Sub procedure inside another Sub procedure or inside a Function procedure.

The `Exit Sub` statement causes the program to immediately exit from a Sub procedure. Execution continues from the statement following the statement that called the Sub procedure. You can write any number of `Exit Sub` statements as required anywhere in a Sub procedure.

Because the Sub procedure does not return a value, Sub procedure names cannot be used in expressions, statements, or commands.

To call a Sub procedure, write the procedure name followed by an argument list. For details on calling Sub procedures, see *7.1.10 Call*.

Two types of variables can be used in Sub procedures: variables that are explicitly declared within the procedure and those that are not. Variables that are explicitly declared in a procedure, using the `Dim` command or equivalent, behave as local variables that are valid only within that procedure. Variables that are used but not explicitly declared in a procedure are also local unless they are explicitly declared at some higher level outside the procedure. The value of local variables in a Sub procedure are discarded when the procedure finishes execution.

A variable that is not explicitly declared in the procedure can be used in a Sub procedure, but it will be handled as a global variable if another variable or entity is defined at the script level with the same name as the undeclared variable.

Example

```
' Call the Sub procedure OperateMessage
' that displays a message dialog box.
Dim Task
Task = "file update"

Call OperateMessage("Start")
    :
Call OperateMessage("Stop")

Sub OperateMessage(status)
    Dim stamp
    stamp = Date + " " + Time
    MsgBox( " [ " & status & " ] " & Task & " at " & stamp)
End Sub
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.10 Call

Purpose

Flow control procedure that passes control to a `Sub` procedure or `Function` procedure.

Syntax

```
[Call] Name [(ArgumentList)]
```

Arguments

`Call`

Optional keyword.

Name

Write the name of the procedure to be called.

ArgumentList

Specify a comma-delimited list of variables or expressions to pass to the procedure.

Description

The `Call` keyword is optional when calling a procedure. However, you must enclose the arguments in parentheses if you call a procedure as the argument of another procedure or command, or if you call a procedure in an expression. If you call a procedure with the `Call` keyword specified, the return value of the called procedure cannot be acquired. To acquire the return value of the procedure, call the procedure with the `Call` keyword omitted.

When a `Function` procedure and `Sub` procedure have the same name, the `Function` procedure takes precedence in a `Call` statement.

Supplementary note

During the execution of a script, the following message might appear: `Could not find the procedure.` In this case, review the naming convention for `Function` and `Sub` procedures and recheck how these procedures are defined.

Examples

```
Call MyProc(0)

MyProc 0
```

JP1/Script version

Supported from JP1/Script 01-00.

7.1.11 Exit xx

Purpose

Flow control statement for exiting from a `While...End loop`, `For...Next loop`, `For...End For loop`, `Do...Loop loop`, `Function` procedure, or `Sub` procedure.

Syntax

```
Exit While

Exit For

Exit Do
```

```
Exit Function
```

```
Exit Sub
```

Description

The purpose of each `Exit` statement is as follows.

`Exit While`

Exits from a `While...End` loop and passes control to the statement following the `End` statement. You can use `Exit While` only within a `While...End` statement. When used within nested `While...End` statements, `Exit While` transfers control to the next loop outside the loop in which it appears.

`Exit For`

Exits from a `For...Next` loop and passes control to the statement following the `Next` statement. You can use `Exit For` only within a `For...Next` or `For...End For` loop. When used within nested `For...Next` or `For...End For` loops, `Exit For` transfers control to the next loop outside the loop in which it appears.

`Exit Do` (from version 05-10)

Exits from a `Do...Loop` statement and passes control to the statement following the `Loop` statement. You can use `Exit Do` only within a `Do...Loop` statement. When used within nested `Do...Loop` statements, `Exit Do` transfers control to the next loop outside the loop in which it appears.

`Exit Function`

Immediately exits from the `Function` procedure in which it appears. Control moves to the statement following the statement that called the `Function` procedure.

`Exit Sub`

Immediately exits from the `Sub` procedure in which it appears. Control moves to the statement following the statement that called the `Sub` procedure.

JP1/Script version

Supported from JP1/Script 01-00.

7.1.12 GoTo

Purpose

Unconditionally jumps to a specified label.

Syntax

```
GoTo LabelName
```

Argument

LabelName

Specify the label name of the jump destination. Use a name that complies with the variable naming conventions.

Description

The `GoTo` statement transfers execution control to a specified label. Execution resumes from the line following the label. A syntax error occurs if multiple labels have the same name at the script level or procedure level.

You can write the label at any position. However, if you specify a `GoTo` statement within a procedure, the labeled line must exist within that procedure.

An execution error occurs if you specify a jump destination label as follows:

- Specify a `GoTo` statement outside the procedure and specify a jump destination label within the procedure.
- Specify a `GoTo` statement outside the flow control statement and specify a jump destination label within the flow control statement.

Example

```
If Exec ("NOTEPAD.EXE" ,True ,"Loging.txt" ) = False Then
  GoTo  ErrorExit  ' Transfer control to ErrorExit.
End
:
ErrorExit:          ' Start processing from here.
:
```

JP1/Script version

Supported from JP1/Script 05-00.

7.1.13 Continue

Purpose

Flow control statement for returning to the start of a `While...End loop`, `For...Next loop`, `For...End For loop`, or `Do...Loop loop`.

Syntax

```
Continue
```

Description

The `Continue` statement can only be used within a `While...End loop`, `For...Next loop`, `For...End For loop`, or `Do...Loop loop`. A syntax error occurs if `Continue` is used in any other position.

You can write any number of `Continue` statements anywhere in these loops. When used within nested loops, `Continue` transfers control to the start of the innermost loop in which it appears.

Example

```
For I = 1 To 10
  For J = 1 To 10  ' Return here from Continue
    ...
    Continue
    ...
  Next
Next
```

JP1/Script version

Supported from JP1/Script 05-00.

7.1.14 On Error

Purpose

The `On Error` statement enables or disables transfer of control to a specified label if an error occurs.

Syntax

```
On Error GoTo LabelName
```

Argument

LabelName

Specify the label name of the jump destination. Use a name that complies with the variable naming conventions.

Specify 0 for *LabelName* to prevent control from transferring to the specified label when an error occurs.

Description

The `On Error` statement enables an error-handling routine starting from the line following the label. If a run-time error occurs, control moves to the labeled line and processing resumes. If a syntax error occurs, however, control is not transferred to the labeled line. The specified label must exist within the same procedure as the `On Error` statement.

If `On Error GoTo 0` is not specified, the error-handling routine starting from the line following the label is effective until the procedure terminates.

An execution error occurs if you specify a jump destination label as follows:

- Specify an `On Error` statement outside the procedure and specify a jump destination label within the procedure.
- Specify an `On Error` statement outside the flow control statement and specify a jump destination label within the flow control statement.

Supplementary note

Control jumps to the label specified in an `On Error` statement if an execution error occurs in a statement or command that can give rise to such errors. As exceptions to this general rule, however, control is not transferred if an execution error results from either of the following:

- End-of-file error in a `TextRead` command
- Timeout in a `WaitForExec` command

For commands that do not give rise to execution errors, such as `IsDefine` and `IsExistFile`, control does not pass to the labeled line regardless of the command's return value.

Example

```
On Error GoTo ErrHandler
    ' Hereafter, if an error occurs,
    ' go to the line following the label.
Copy (InDir+"CTL3D32.DLL", OutDir+"CTL3D32.DLL",
    VersionUp)
...
On Error GoTo 0 ' Hereafter, do not transfer control
    ' to the line following the label.
...
ErrHandler:    ' Error-handling jump destination
    ...
```

JP1/Script version

Supported from JP1/Script 05-00.

8

Basic Commands

This chapter describes the basic commands you can use when creating a script.

8.1 List of basic commands

Table 8-1 lists the basic commands that you can use when creating a script. The numbers enclosed in parentheses in the *Command* column identify the version with which the command was first introduced (if there is no version number shown, the command has been supported since version 01-00).

Table 8–1: List of basic commands

Category	Command	Description
Variable manipulation	Dim	Declares a variable and allocates space in memory.
	Dim (Array) (06-00)	Declares an array variable and allocates space in memory.
	SetEnvironment or SetEnv	Sets an environment variable.
	GetEnvironment or GetEnv	Gets an environment variable.
	SetGV	Sets a global variable on the local or remote PC.
	GetGV	Gets a global variable on the local or remote PC.
	DeleteGV (06-00)	Deletes a global variable on the local or remote PC.
	GetArrayCount (06-00)	Counts the elements in an array variable (or the number of rows or columns in a two-dimensional array).
String manipulation	InStr (05-00)	Searches for a specified substring in a string and returns the character position (offset from the beginning) of the first occurrence.
	InArray (06-00)	Searches for a specified string among the elements in an array variable, and returns the index number of the first occurrence.
	Len	Returns the length of a string.
	Lcase	Converts uppercase characters in a string to lowercase.
	Ucase	Converts lowercase characters in a string to uppercase.
	Left	Returns a specified number of characters from the left side of a string.
	Mid	Returns a specified number of characters from inside a string.
	Right	Returns a specified number of characters from the right side of a string.
	Space	Returns a string containing a specified number of spaces.
	Ltrim (05-00)	Returns a string without leading spaces.
	Rtrim (05-00)	Returns a string without trailing spaces.
	Trim (05-00)	Returns a string without leading or trailing spaces.
	+ (string concatenation)	Performs string concatenation on two expressions.
	& (string concatenation) (06-00)	Performs string concatenation on two expressions.
	&= (string concatenation) (06-51)	Performs string concatenation on a variable and an expression, and assigns the result to the variable.
AddStr (05-00)	Returns a concatenated string consisting of two or more strings with specified delimiters inserted.	

Category	Command	Description
String manipulation	SeparateStrCount (05-00)	Returns the number of strings split at a specified delimiter.
	SeparateStr (05-00)	Returns a string split at a specified delimiter.
	Str (05-20)	Returns a specified value as a string.
	Format (05-20)	Returns a specified value as a formatted string.
	IsLower (06-00)	Checks whether a string is lowercase characters, and returns <code>True</code> or <code>False</code> .
	IsUpper (06-00)	Checks whether a string is uppercase characters, and returns <code>True</code> or <code>False</code> .
	IsSingleChar (06-00)	Checks whether a string is one-byte characters, and returns <code>True</code> or <code>False</code> .
	IsMultiChar (06-00)	Checks whether a string is two-byte characters, and returns <code>True</code> or <code>False</code> .
Working with dates	Date	Returns the current date.
	Time	Returns the current time.
	Year	Returns a four-digit number representing the year for a specified date.
	Month	Returns a two-digit number in the range 1 to 12, representing the month for a specified date.
	Day	Returns a two-digit number in the range 1 to 31, representing the day for a specified date.
	Weekday	Returns a one-digit number in the range 1 (Sunday) to 7 (Saturday), representing the day of the week for a specified date.
	Hour	Returns a two-digit number in the range 0 to 23, representing the hour for a specified time.
	Minute	Returns a two-digit number in the range 0 to 59, representing the minute for a specified time.
	Second	Returns a two-digit number in the range 0 to 59, representing the second for a specified time.
	CalcDate (05-20)	Adds or subtracts a specified interval to or from a specified date.
	CompDate (05-20)	Compares two specified dates and returns <code>True</code> or <code>False</code> .
	GetDateCount (05-20)	Calculates the difference between two dates.
	CalcTime (05-20)	Adds or subtracts a specified interval to or from a specified time.
	CompTime (05-20)	Compares two specified times and returns <code>True</code> or <code>False</code> .
	GetTimeCount (05-20)	Calculates the difference between two times.
IsLeapYear (05-20)	Checks whether a specified year is a leap year and returns <code>True</code> or <code>False</code> .	
File and folder management	IniRead	Reads data from an initialization file.
	IniWrite	Sets data in an initialization file.

Category	Command	Description
File and folder management	TextFileReplace	Replaces a specified string in a text file.
	TextOpen (05-00)	Opens a text file.
	TextClose (05-00)	Closes a text file.
	TextRead (05-00)	Reads one line of data from a text file.
	TextWrite (05-00)	Writes data to a text file.
	TextSeek (05-00)	Moves the read/write position in a text file.
	GetTextPosition (05-00)	Returns the current read/write position in a text file.
	MakeDir	Creates a folder.
	DeleteDir	Deletes a folder.
	DeleteFile	Deletes a file.
	Rename	Renames a file.
	TempDir	Gets a temporary folder name.
	TempFile	Creates a temporary file.
	SetFileAttribute or SetFileAttr	Sets file attributes.
	GetFileAttribute or GetFileAttr (06-00)	Gets file attributes.
	SetFileTime (05-10)	Sets the date and time of a file.
	GetFileTime (05-10)	Gets the date and time of a file.
	GetFileSize (06-00)	Gets the size of a file.
	GetVersionInfo or GetVerInfo (06-00)	Gets version information for a file.
	SplitFile (05-10)	Partitions a file by a specified size.
	CatFiles (05-10)	Joins split files into one file.
	SetStandardFile or SetStdFile (05-10)	Sets the standard input, standard output, and standard error files for processes called by the EXEC command.
	ResetStandardFile or ResetStdFile (05-10)	Resets the standard input, standard output, and standard error files for processes called by the EXEC command.
	SplitPath	Analyzes a full path.
	MakePath	Creates a full path.
	SetPath	Sets a folder path to the executable folder.
	GetPath	Gets the path to the executable folder.
	SetVolumeLabel or SetVolLabel (05-10)	Sets the volume label of a disk.
	GetVolumeLabel or GetVolLabel (05-10)	Gets the volume label of a disk.
	GetDiskFreeSpace (05-10)	Gets the amount of free space on a disk.
	Copy	Copies files.

Category	Command	Description
Message output	InputBox	Displays a message and entry boxes in a dialog box, and returns the entry box contents when the user enters text or clicks a button.
	Message	Outputs specified text to a file or window. Also erases the displayed window and message text.
	MessageBox	Displays a specified message in a dialog box. Buttons or icons may be added. When the box has buttons, the command returns a value indicating which button the user clicked.
	MessageEventLog (01-01)	Outputs a message to the application log in Event Viewer.
	IMEventMessage (05-20)	Issues events to JP1/IM or JP1/Base. For details about the function of JP1/IM event console, see the manual <i>JP1/Integrated Management - Manager Overview and System Design Guide</i> .
Menu display	Menu (05-20)	Displays a user-defined window (menu form).
Calculations	+ operator (addition)	Finds the sum of two expressions.
	+= operator (addition) (06-51)	Adds an expression to a variable, and assigns the sum to the variable.
	- operator (subtraction)	<ul style="list-style-type: none"> Finds the difference between two numbers. Reverses the sign of a numerical expression.
	-= operator (subtraction and negation) (06-51)	Subtracts an expression from a variable, and assigns the difference to the variable.
	Mod operator (remainder division)	Divides one number by another and returns the remainder.
	Mod= operator (remainder division) (06-51)	Divides a variable by an expression, and assigns the remainder to the variable.
	* operator (multiplication)	Finds the product of two numbers.
	*= operator (multiplication) (06-51)	Multiplies an expression by a variable, and assigns the result to the variable.
	/ operator (division)	Finds the quotient of two numbers and returns an integer.
	/= operator (division) (06-51)	Divides a variable by an expression, and assigns the integer part to the variable.
	\ operator (integer division)	Finds the quotient of two numbers and returns an integer.
	\= operator (integer division) (06-51)	Divides a variable by an expression, and assigns the integer part to the variable.
	^ operator (exponentiation) (06-00)	Raises one number to the power of another.
	^= operator (exponentiation) (06-51)	Finds the exponentiation of a variable and an expression, and assigns the result to the variable.
	Comparison operators (=, <>, <, <=, >, and >=)	Compares two expressions.
	And operator (logical AND)	Finds the logical AND of two expressions.
Or operator (logical OR)	Finds the logical OR of two expressions.	
Not operator (logical NOT) (06-00)	Finds the logical NOT of an expression.	
Evaluations	IsEmpty	Checks whether a variable is an Empty value and returns True or False.

Category	Command	Description
Evaluations	IsDefine or IsDef	Checks whether a variable is defined and returns True or False.
	IsNumeric (01-01)	Checks whether a value can be evaluated as a numeric and returns True or False.
	IsEmptyDir	Checks whether a folder is empty and returns True or False.
	IsExistDir	Checks whether a folder exists and returns True or False.
	IsExistFile	Checks whether a file exists and returns True or False.
	IsWriteableDir	Checks whether a folder is writeable and returns True or False.
	IsFileAttribute or IsFileAttr	Checks whether a folder or file has a specific attribute and returns True or False.
	IsNew (05-10)	Compares which of two files has the more recent version or date, and returns True or False.
	CheckDirName	Checks whether a folder name ends with a backslash (\).
	CheckDriveType	Checks the drive type.
External program calls	Exec [#]	Calls an executable file. Multiple parameters may be specified.
	NetExec (05-00) [#]	Calls an executable file on the local or remote PC. Multiple parameters may be specified.
	WaitForExec (01-01)	Waits for completion or forcibly terminates an executable file called by the Exec or NetExec command.
	GetExecStatus (06-00)	Gets the current execution status of an executable file called by the Exec or NetExec command.
	CallSpt (06-71) [#]	Calls an SPT file as a procedure within the current process. Multiple parameters may be specified.
Automatic start	EntryStartUp (05-10)	Registers a script file for automatic startup.
	CancelStartUp (05-10)	Cancels automatic startup for a script file.
Comment	Rem or ' (single quote)	Indicates a comment line in a program.
Other commands	Sleep	Halts script execution for a specified time.
	Alert	Displays or clears user error icons in the Trace Viewer status bar.
	Beep	Sounds a beep from the speakers. See <i>1.7.4 About the Beep command</i> and the note in <i>8.13.3 Beep (sound a beep from the speakers)</i> for details about using the Beep command.
	Exit	Terminates script execution.
	GetErrorMessage (01-01)	Returns the error message associated with a specified error detail code.

[#]: Table 8-2 shows the differences between the Exec, NetExec, and CallSpt commands.

Table 8–2: Differences between the Exec, NetExec, and CallSpt commands

Command	Executable files	Wait for execution	Called external file process		Purpose
			Feature	Environment variable	
Exec	EXE file, BAT file, COM file, SPT file, CMD file, and linked files	User can specify whether to wait for execution.	Executed in a new process.	The new process inherits a copy of the calling process and can therefore recognize the value of the environment variable at that process. However, any update of the variable is not reflected back at the calling process.	For calling an executable file.
NetExec	EXE file, BAT file, COM file, SPT file, CMD file, and linked files	User can specify whether to wait for execution.	Executed in a new process.	The called executable file does not inherit the process environment variables of the calling process. Therefore, the values set in the calling side cannot be referenced.	For calling an executable file on a remote computer, or for calling an executable file and specifying whether it is to be executed in the Service space or logon space.
CallSpt	SPT file	Not specifiable (waits for execution by default).	Executed in the current process.	The environment variable is shared with the calling process and can be referenced. The calling process can also reference the updated value.	For calling an executable file as part of the current process.

8.2 Commands for manipulating variables

8.2.1 Dim (declare a variable and allocate space)

Purpose

Declares a variable and allocates space in memory.

Syntax

```
Dim VarName [, VarName, ...]
```

Arguments

VarName

Specify the name of the variable you are defining. Comply with the variable naming conventions.

Description

A variable declared at the script level using the `Dim` command can be referenced within that script by all procedures. A variable declared at the procedure level can be referenced only within that procedure.

Note that a variable allocated by assignment to a variable declared or undeclared in the `Do...Loop`, `For...Next`, `For...End For`, or `While...End` loop in the script or procedure can be referenced only within the loop.

If a value is assigned to a variable that has not been declared by using the `Dim` command, the system automatically declares the variable, allocates a memory area, and then assigns the value.

Once declared, a variable has an `Empty` value.

JP1/Script version

Supported from JP1/Script 01-00.

8.2.2 Dim (array) (declare an array variable and allocate space)

Purpose

Declares an array variable and allocates space in memory.

Syntax

```
Dim VarName ([Number1] [, Number2]) [, VarName ([Number1] [,  
Number2]), ...]  
Dim VarName [ [Number1] [, Number2] ] [, VarName [ [Number1] [,  
Number2] ], ...]
```

Arguments

VarName

Specify the variable name of the array variable you are defining. Comply with the variable naming conventions.

Number1

Specify the number of elements in the array variable (or the number of rows for a two-dimensional array). Write a number or a variable that stores this value.

If you omit this argument, the array variable will have a variable number of elements.

Number2

For a two-dimensional array variable, specify the number of columns or a variable that stores this value.

If you omit this argument, the array variable will have a variable number of elements.

Description

An array variable declared at the script level using the `Dim (array)` command can be referenced within that script by all procedures. An array variable declared at the procedure level can be referenced only within that procedure.

Note that an array variable declared in a `Do...Loop`, `For...Next`, `For...End For`, or `While...End loop` at the script level or procedure level can be referenced only within that loop.

When a fixed array variable is declared, the `Empty` value is set for all the elements of the array variable.

Examples

```
' Declare table1 as a fixed two-dimensional array
' variable.
Dim table1(5,6)

' Assign all the elements of the fixed one-dimensional
' array variable table2 to the dynamic one-dimensional
' array variable table3.
Dim table2(6), table3()
For i = 1 To 6
    table2(i) = Time()
Next
table3 = table2

' Compare the fixed one-dimensional array variable table4
' with array variable table5.
Dim table4(6), table5(6)
...
If table4 = table5 Then
    ...
' The following size comparison is not allowed and results
' in an error.
If table4 <= table5 Then
    ...
```

JP1/Script version

Supported from JP1/Script 06-00.

8.2.3 SetEnvironment or SetEnv (set an environment variable)

Purpose

Sets an environment variable.

Syntax

```
SetEnvironment (Type, EnvironmentName [, Value])
SetEnv (Type, EnvironmentName [, Value])
```

Arguments

Type

Specify whether the environment variable you are defining is a system environment variable, user environment variable, or an environment variable for the current process.

Use one of the following values:

Value	Meaning
SystemEnv	System environment variable Takes effect at system restart after execution of this command. Can be obtained after command execution using the <code>GetEnvironment</code> command.
UserEnv	User environment variable Takes effect at system restart after execution of this command. Can be obtained after command execution using the <code>GetEnvironment</code> command.
ProcessEnv	Environment variable for the current process Takes effect only within the current process after execution of this command.

If you specify `SystemEnv`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

EnvironmentName

Write the name of the environment variable as a character string or as a variable that stores this value.

Value

Write the value of the environment variable as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a zero-length string ("") is assumed.

If the value you specify contains a reference to the unexpanded environment variable ("%PATH%", for example), the value is still set as specified. For example, if you set "%PATH%;C:\Data" in *Value*, "%PATH%;C:\Data" is set as the value of the environment variable. However, when this environment variable is referenced, the system obtains a value in which the %PATH% portion is replaced by the current value of the PATH environment variable. To specify a process environment variable, which cannot be replaced with the current value, code the script as shown in Example 1. If the system cannot find a PATH environment variable, it gets the "%PATH%" portion as is.

Description

The `SetEnvironment` or `SetEnv` command sets a specified value in a specified environment variable. The command returns `True` on successful execution, or `False` if an error occurs.

If you specify a non-existent environment variable, a new environment variable is created with the specified value. If you specify an existing environment variable, its value is updated.

Example 1

```
' Add "C:\ABC\" to the PATH process environment variable.
Dim buff1
buff1 = GetEnvironment ( ProcessEnv , "PATH" )
SetEnvironment ( ProcessEnv , "PATH" , buff1 & ";C:\ABC\" )
Exec ( "ABC.EXE" , True )
```

Example 2

```
' Set the PATH environment variable to the APP_PATH system
' environment variable.
SetEnvironment ( SystemEnv , "APP_PATH" , "%PATH%" )
```

JP1/Script version

Supported from JP1/Script 01-00.

8.2.4 GetEnvironment or GetEnv (get an environment variable)

Purpose

Gets an environment variable.

Syntax

```
GetEnvironment (Type, EnvironmentName [, Option])  
GetEnv (Type, EnvironmentName [, Option])
```

Arguments

Type

Specify whether the environment variable you are defining is a system environment variable, user environment variable, or an environment variable for the current process.

Use one of the following values:

Value	Meaning
SystemEnv	System environment variable
UserEnv	User environment variable
ProcessEnv	Environment variable for the current process

EnvironmentName

Write the name of the environment variable as a character string or as a variable that stores this value.

Option (from version 05-20)

Specify the following option:

Value	Meaning
Expand	If the value to be obtained includes a reference to an unexpanded environment variable ("%PATH%", for example), get the expanded value.

Description

The `GetEnvironment` command gets a specified environment variable and returns its value as the execution result.

If you specify a non-existent environment variable, a zero-length string ("") is returned.

Note

If the value of the specified environment variable is a character string containing 1,025 or more bytes, the command returns the first 1,024 bytes as the execution result. The 1,025th and subsequent bytes are ignored.

Examples

```
' If environment variable path has the value "C:\Winnt",  
' this code stores "C:\Winnt" in variables buff1 and  
' buff2.  
Dim buff1, buff2  
buff1 = GetEnvironment (SystemEnv, "path")  
buff2 = GetEnvironment (SystemEnv, "path", Expand)  
  
' If environment variable path has the value  
' "%SystemRoot%" and environment variable SystemRoot has  
' the value "C:\Winnt", this code stores "%SystemRoot%"  
' in variable buff3, and "C:\Winnt" in variable buff4.  
Dim buff3, buff4
```

```
buff3 = GetEnvironment (SystemEnv, "path")
buff4 = GetEnvironment (SystemEnv, "path", Expand)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.2.5 SetGV (set a global variable)

Purpose

Sets a global variable. In JP1/Script 05-00 and later versions, this command allows you to set a global variable on a remote computer.

Syntax

```
SetGV (GlobalName, [Value] [, CompName])
```

Arguments

GlobalName

Write the global variable name as a character string or as a variable that stores this value.

Value

Write the value to be set in the global variable as a string, number, or as a variable that stores this value.

This value is optional. If you omit this value, a zero-length string ("") is assumed.

CompName (from version 05-00)

Write the name of the computer on which this global variable is to reside, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the computer on which the command is executed is assumed.

Note that JP1/Script must be installed and the JP1/Script service must be running on the remote computers to be specified. For details about accounts for accessing computers, see [2.2 Accounts for communication with other computers](#).

Description

The SetGV command sets a specified value in a specified global variable. The command returns `True` on successful execution, or `False` if an error occurs.

If you specify a non-existent global variable, a new global variable is created with the specified value. If you specify an existing global variable, its value is updated.

If you specify a remote computer in *CompName* and want to include *n* double quotation marks (") as characters in either *GlobalName* or *Value*, you must specify *n**4 double quotation marks (") for *GlobalName* and *n**8 double quotation marks (") for *Value*. However, if you specify 1 in the following registry, you can use *n**2 double quotation marks instead of *n**8:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
ParseDQ
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify *n**2 double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify *n**2 double quotation marks (") is enabled.

To specify both 1 and 2 (see the `SeparateStrCount` and `SeparateStr` commands) as the value of this registry, type 3.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note

This command creates a global variable file (`SPTGV.SPG`) in the `DATA` folder in the installation folder. However, in JP1/Script 06-51 and later versions, if you open the Options (Cluster Environment) dialog box (**Tools, Options** command) and change the folder for management file output, the file will be created in the folder you specified.

The global variable file is preserved when the script completes execution. Delete this file if you want to initialize the global variables.

When a remote computer is specified in the *CompName* argument, operation is guaranteed only in a LAN environment. This command does not operate across firewalls.

Examples

```
' Set a global variable on the local computer.
SetGV ("Debug_Mode", True)

' Set a global variable on computer "SOP4A065 (SCRIPT)".
SetGV ("Debug_Mode", True, "SOP4A065 (SCRIPT)")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.2.6 GetGV (get a global variable)

Purpose

Gets a global variable. In JP1/Script 05-00 and later versions, this command allows you to get a global variable on a remote computer.

Syntax

```
GetGV (GlobalName [, CompName])
```

Arguments

GlobalName

Write the global variable name as a character string or as a variable that stores this value.

CompName (from version 05-00)

Write the name of the computer on which this global variable resides, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the computer on which the command is executed is assumed.

Note that JP1/Script must be installed and the JP1/Script service must be running on the remote computers to be specified. For details about accounts for accessing computers, see [2.2 Accounts for communication with other computers](#).

Description

The `GetGV` command acquires a specified global variable and returns its value. If you specify a non-existent global variable, a zero-length string ("") is returned.

If you specify a remote computer in *CompName* and want to include *n* double quotation marks (") as characters in *GlobalName*, you must enter *n*4* double quotation marks. However, if you specify 1 in the following registry, you can use *n*2* double quotation marks instead of *n*8*:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
ParseDQ
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") is enabled.

To specify both 1 and 2 (see the `SeparateStrCount` and `SeparateStr` commands) as the value of this registry, type 3.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note

When a remote computer is specified in the *CompName* argument, operation is guaranteed only in a LAN environment. This command does not operate across firewalls.

Example

```
' Get a global variable on the local computer and write
' to the registry.
If GetGV ("Debug_Mode") = True Then
    Dbflag = "DebugFlag"
    RegWrite (HKEY_CURRENT_USER
              , "Software\Hitachi\Script\Option", Dbflag, True)
End

' Get and update the global variable on computer
' "SOP4A065 (SCRIPT)".
If GetGV ("Debug_Mode", "SOP4A065 (SCRIPT)") = True Then
    SetGV ("Debug_Mode", False, "SOP4A065 (SCRIPT)")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.2.7 DeleteGV (delete a global variable)

Purpose

Deletes a global variable on the local computer or on a remote computer.

Syntax

```
DeleteGV (GlobalName [, CompName])
```

Arguments

GlobalName

Write the global variable name as a character string or as a variable that stores this value. Alternatively, you can specify the following value:

Value	Meaning
AllGV	Delete all global variables, but preserve the global variable file (SPTGV.SPG).

CompName

Write the name of the computer from which to delete this global variable, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the computer on which the command is executed is assumed.

Note that JP1/Script must be installed and the JP1/Script service must be running on the remote computers to be specified. For details about accounts for accessing computers, see [2.2 Accounts for communication with other computers](#).

Description

The DeleteGV command deletes a specified global variable. The command returns `True` on successful execution, or `False` if an error occurs.

If you specify a non-existent global variable, the command returns `True` without doing anything.

If you specify a remote computer in *CompName* and want to include *n* double quotation marks (") as characters in *GlobalName*, you must enter *n*4* double quotation marks. However, if you specify 1 in the following registry, you can use *n*2* double quotation marks instead of *n*8*:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
ParseDQ
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify *n*2* double quotation marks (") is enabled.

To specify both 1 and 2 (see the `SeparateStrCount` and `SeparateStr` commands) as the value of this registry, type 3.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note

When a remote computer is specified in the *CompName* argument, operation is guaranteed only in a LAN environment. This command does not operate across firewalls.

Examples

```
' Delete all global variables on the local computer.
DeleteGV (AllGV)

' Delete global variable "prm01" on computer
' "SOP4A065 (SCRIPT)".
SetGV ("prm01", 10, "SOP4A065 (SCRIPT)")
Exec (_SCF_"ABC.SPT", True)
DeleteGV ("prm01", "SOP4A065 (SCRIPT)")
```

JP1/Script version

Supported from JP1/Script 06-00.

8.2.8 GetArrayCount (count the elements in an array variable)

Purpose

Counts the elements in an array variable (or the number of rows or columns in a two-dimensional array).

Syntax

```
GetArrayCount (ArrayName)
```

Argument

ArrayName

Specify the array variable as a variable name.

Description

The `GetArrayCount` command gets the number of elements in a specified array variable (or the number of rows or columns in a two-dimensional array). On successful completion, the command returns the number of elements, or the number of rows or columns for a two-dimensional array. If an error occurs, the command returns a zero-length string ("").

If you specify a dynamic array variable, the `GetArrayCount` command returns the number of elements (or rows or columns) already set.

Examples

```
' A is a fixed one-dimensional array variable. This code
' stores the number of elements (10) in variable result1.
Dim A(10)
result1 = GetArrayCount (A)

' B is a fixed two-dimensional array variable. This code
' stores the number of columns (10) in variables result2
' and result3, and the number of rows (5) in variable
' result4.
Dim B(5, 10)
result2 = GetArrayCount (B(1))
result3 = GetArrayCount (B(5))
result4 = GetArrayCount (B)
```

```
' C is a dynamic two-dimensional array variable. This code
' stores the number of columns (1 and 7) in variables
' result5 and result6, and the number of rows (2) in
' variable result7.
Dim C(,)
C(2,1) = "SUN"
C(2,2) = "MON"
C(2,3) = "TUE"
C(2,4) = "WED"
C(2,5) = "THU"
C(2,6) = "FRI"
C(2,7) = "SAT"
result5 = GetArrayCount (C(1))
result6 = GetArrayCount (C(2))
result7 = GetArrayCount (C)
```

JP1/Script version

Supported from JP1/Script 06-00.

8.3 Commands for manipulating strings

8.3.1 InStr (find the position of a substring in a string)

Purpose

Searches for a specified substring in a specified string and returns the character position (number of characters from the beginning of the string) of the first occurrence of the substring.

Syntax

```
InStr (String, SearchStr, [Start] [, Compare])
```

Arguments

String

Write the string to search as a character string or as a variable that stores this value.

Zero is returned if you specify a zero-length string ("").

SearchStr

Write the search substring as a character string or as a variable that stores this value.

Zero is returned if you specify a zero-length string ("").

Start

Specify the position at which to begin the search as the offset from beginning of the string you set in *String*, where 1 is the first character.

Zero is returned if the *Start* value exceeds the number of characters in *String*.

This value is optional. If you omit this value, 1 is assumed.

Compare

Specify how to perform string comparisons. Use one of the following values:

Value	Meaning
True	Perform case-sensitive comparison.
False	Perform case-insensitive comparison.
Twice	Perform case-insensitive comparison for one-byte characters, and case-sensitive comparison for two-byte characters.

This value is optional. If you omit this value, *Twice* is assumed.

Description

The `InStr` command searches for a specified substring in a specified string and returns the character position (number of characters from the beginning) of the first occurrence. One two-byte-character and one one-byte-character are both handled as one character. Zero is returned if no matching string is found.

Example

```
' This code stores 16 in variable point1.  
Dim point1  
point1 = InStr ("Search file in ABC order", "abc", 3, False)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.2 InArray (find the position of a string in an array variable)

Purpose

Searches for a specified string among the elements in a specified array variable, and returns the index number of the first occurrence.

Syntax

```
InArray (ArrayName, SearchStr, [Start] [, Compare])
```

Arguments

ArrayName

Specify the array variable to search as a variable name.

Zero is returned if the specified array variable has no elements.

SearchStr

Write the search string as a character string or as a variable that stores this value.

Zero is returned if you specify a zero-length string ("").

Start

Specify the position in the specified array variable at which to begin the search. Specify the start position as an index number or as a variable that stores this value.

Zero is returned if the start position exceeds the number of elements in the array variable set in *ArrayName*.

This value is optional. If you omit this value, the first element is assumed.

Compare

Specify how to perform string comparisons. Use one of the following values:

Value	Meaning
True	Perform case-sensitive comparison.
False	Perform case-insensitive comparison.
Twice	Perform case-insensitive comparison for one-byte characters, and case-sensitive comparison for two-byte characters.

This value is optional. If you omit this value, *Twice* is assumed.

Description

The `InArray` command searches for a specified string among the elements of a specified string and returns the index number (number from 1) of the first occurrence. Zero is returned if no matching string is found.

Example 1

Search for specified data from the array variable `closeDay`, which contains the following values:

	(Column 1)	(Column 2)	(Column 3)	...	(Column 11)	(Column 12)
(Row 1)	"January"	"February"	"March"	...	"November"	"December"
(Row 2)	5	2	1	...	1	6
(Row 3)	19	16	15	...	15	20

```
Dim closeDay(3,12)
...
' (Store the values in array variable closeDay.)
...
monthName = "March"
buff = InArray (closeDay(1), monthName, 1, False)
If buff > 0 Then
```

```

firstDay = closeDay(2, buff)
MessageBox ("First non-working day of" + monthName + _
           "is" + firstDay + ".")
End If

```

Example 2

```

' Search for specified data among all the elements of a
' two-dimensional array variable.
Dim array1(,)
...
' (Store the values in array variable array1.)
...
allCnt = GetArrayCount (array1)
For line = 1 To allCnt
  buff = InArray (array1(line), "1999", 1, False)
  If 0 < buff Then
    Exit For
  End If
Next

```

JP1/Script version

Supported from JP1/Script 06-00.

8.3.3 Len (calculate the length of a string)

Purpose

Returns the length of a string.

Syntax

```
Len (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

Description

The Len command returns the number of characters in a string, or the number of characters in the string stored in a specified variable. One two-byte-character and one one-byte-character are both handled as one character.

Example

```

' This code stores 6 in variable length1.
Dim length1
length1 = Len ("ABCDEF")

```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.4 LCase (convert one-byte alphabetic characters to lowercase)

Purpose

Converts the one-byte uppercase alphabetic characters in a string to lowercase, and returns the converted character string.

Syntax

```
LCase (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

Description

The LCase command converts the one-byte uppercase alphabetic characters in a string to lowercase, and returns the converted character string.

Characters in the string other than one-byte uppercase alphabetic characters are not affected.

Example

```
' This code stores "abcdef" in variable string1.  
Dim string1  
string1 = LCase ("abcDEF")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.5 UCase (convert one-byte alphabetic characters to uppercase)

Purpose

Converts the one-byte lowercase alphabetic characters in a string to uppercase, and returns the converted character string.

Syntax

```
UCase (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

Description

The UCase command converts the one-byte lowercase alphabetic characters in a string to uppercase, and returns the converted character string.

Characters in the string other than lowercase characters are not affected.

Example

```
' This code stores "ABCDEF" in variable string1.  
Dim string1  
string1 = UCase ("abcDEF")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.6 Left (return characters from the left side of a string)

Purpose

Returns a substring containing a specified number of characters from the left side of a string.

Syntax

```
Left (String, Length)
```

Arguments

String

Specify the string itself or a variable that stores this value.

Length

Specify the number of characters to retrieve from the specified string, or write a variable that stores this value.

If you specify 0, a zero-length string ("") is returned. If you specify a value greater than all the characters in the string you set in *String*, the `Left` command returns all the characters in the specified string.

Description

The `Left` command returns a substring containing a specified number of characters from the left side of a specified string.

One two-byte-character and one one-byte-character are both handled as one character. To check the length of the string, use the `Len` command.

Example

```
' This code stores "ABCD" in variable string1.  
Dim string1  
string1 = Left ("ABCDEFGH", 4)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.7 Mid (return characters from inside a string)

Purpose

Returns a substring containing a specified number of characters from inside a string.

Syntax

```
Mid (String, Start [, Length])
```

Arguments

String

Specify the string itself or a variable that stores this value.

Start

Specify the position from which to retrieve the characters as the offset from beginning of the string you set in *String*, where 1 is the first character.

A zero-length string ("") is returned if the *Start* value exceeds the number of characters in *String*.

Length

Specify the number of characters to retrieve, or specify a variable that stores this value.

If you omit this value, or if the string contains fewer characters than this value, the `Mid` command returns all the characters from the start position.

Description

The `Mid` command returns a substring containing a specified number of characters from inside a specified string. One two-byte-character and one one-byte-character are both handled as one character.

To check the length of the string, use the `Len` command.

Example

```
' This code stores "CDEFG" in variable string1.  
Dim string1  
string1 = Mid ("ABCDEFGH", 3, 5)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.8 Right (return characters from the right side of a string)

Purpose

Returns a substring containing a specified number of characters from the right side of a string.

Syntax

```
Right (String, Length)
```

Arguments

String

Specify the string itself or a variable that stores this value.

Length

Specify the number of characters to retrieve from the specified string. Write the number itself or a variable that stores this value.

If you specify 0, a zero-length string ("") is returned. If you specify a value greater than all the characters in the string you set in *String*, the `Right` command returns all the characters in the specified string.

Description

The `Right` command returns a substring containing a specified number of characters from the right side of a specified string.

One two-byte-character and one one-byte-character are both handled as one character.

To check the length of the string, use the `Len` command.

Example

```
' This code stores "EFGH" in variable string1.
Dim string1
string1 = Right ("ABCDEFGH", 4)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.9 Space (return a specified number of one-byte spaces)

Purpose

Returns a string containing a specified number of one-byte spaces.

Syntax

```
Space (Number)
```

Argument

Number

Specify the number of spaces or a variable that stores this value. The specified range is zero to 1,024. If you specify 0, a zero-length string ("") is returned. If you specify a value greater than 1,024, 1,024 is assumed.

Description

The Space command returns a string containing a specified number of one-byte spaces.

Example

```
' Store three spaces in variable string1.
Dim string1
string1 = Space (3)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.10 LTrim (remove leading spaces from a string)

Purpose

Returns a string with the leading spaces removed.

Syntax

```
LTrim (String [, Option])
```

Arguments

String

Specify the string itself or a variable that stores this value.

If no leading spaces are found in the specified string, the command returns the entire string.

Option

Specify the following optional value:

Value	Meaning
Twice	Remove all leading spaces.

Description

If you omit the *Option* argument, the `LTrim` command returns a string with one-byte leading spaces removed.

If you specify `Twice` in *Option*, both one-byte and two-byte leading spaces are removed.

Example

```
' Store "ABC DEFG  " in variable string1.
Dim string1
string1 = LTrim ("  ABC DEFG  ", Twice)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.11 RTrim (remove trailing spaces from a string)

Purpose

Returns a string with the trailing spaces removed.

Syntax

```
RTrim (String [, Option])
```

Arguments

String

Specify the string itself or a variable that stores this value.

If no trailing spaces are found in the specified string, the command returns the entire string.

Option

Specify the following optional value:

Value	Meaning
Twice	Remove all trailing spaces.

Description

If you omit the *Option* argument, the `RTrim` command returns a string with one-byte trailing spaces removed.

If you specify `Twice` in *Option*, both one-byte and two-byte trailing spaces are removed.

Example

```
' Store "  ABC DEFG" in variable string1.
Dim string1
string1 = RTrim ("  ABC DEFG  ", Twice)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.12 Trim (remove leading and trailing spaces from a string)

Purpose

Returns a string with both leading and trailing spaces removed.

Syntax

```
Trim (String [, Option])
```

Arguments

String

Specify the string itself or a variable that stores this value.

If no leading or trailing spaces are found in the specified string, the command returns the entire string.

Option

Specify the following optional value:

Value	Meaning
Twice	Remove all trailing spaces.

Description

If you omit the *Option* argument, the Trim command returns a string with one-byte leading and trailing spaces removed.

If you specify Twice in *Option*, both one-byte and two-byte leading and trailing spaces are removed.

Example

```
' Store "ABC DEFG" in variables string1, string2, and
' string3.
Dim string1, string2, string3
string1 = Trim ("  ABC DEFG  ", Twice)
string2 = Trim ("      ABC DEFG", Twice)
string3 = Trim ("ABC DEFG      ", Twice)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.13 + operator (concatenate strings)

Purpose

Performs string concatenation on two expressions.

Syntax

```
Result = Expression1 + Expression2
```

Arguments

Result

Specify a variable for storing the result.

Expression1

Write any expression.

Expression2

Write any expression.

Description

The action of the + operator depends on the internal processing of the two expressions, as follows.

Condition	Operation performed
Both expressions are strings.	String concatenation
Both expressions are numbers.	Addition
Both expressions are strings consisting entirely of numbers.	Addition [#]
One expression is a number, and the other is a string.	String concatenation
One expression is a string, and the other is a string consisting entirely of numbers.	String concatenation
One expression is a number, and the other is a string consisting entirely of numbers.	Addition

If both expressions are Empty values, the number 0 is set in *Result*. However, if only one of the expressions is an Empty value, the other expression is returned as is in *Result*.

#

Use the & operator to concatenate strings.

Note

When *Result* and *Expression1* are the same variable, you can use the += operator instead.

Examples

```
' This codes stores "ABCDEF" in variable result1.  
result1 = "ABC" + "DEF"  
  
' This codes stores 12 in variable result2.  
result2 = 7 + 5
```

JP1/Script version

Supported from JP1/Script 01-00.

8.3.14 & operator (concatenate strings)

Purpose

Performs string concatenation on two expressions.

Syntax

```
Result = Expression1 & Expression2
```

Arguments

Result

Specify a variable for storing the result.

Expression1

Write any expression.

Expression2

Write any expression.

Description

The action of the & operator depends on the internal processing of the two expressions, as follows.

Condition	Operation performed
Both expressions are characters.	String concatenation
Both expressions are numbers.	String concatenation
Both expressions are strings consisting entirely of numbers.	String concatenation
One expression is a number, and the other is a string.	String concatenation
One expression is a string, and the other is a string consisting entirely of numbers.	String concatenation
One expression is a number, and the other is a string consisting entirely of numbers.	String concatenation

If both expressions are Empty values, a zero-length string ("") is set in *Result*. However, if only one of the expressions is an Empty value, the other expression is returned as is in *Result*.

Note

When *Result* and *Expression1* are the same variable, you can use the &= operator instead.

Examples

```
' This codes stores "ABCDEF" in variable result1.  
result1 = "ABC" & "DEF"  
  
' This codes stores 75 in variable result2.  
result2 = 7 & 5
```

JP1/Script version

Supported from JP1/Script 06-00.

8.3.15 &= operator (concatenate strings)

Purpose

Performs string concatenation on a variable and an expression, and assigns the result to the variable.

Syntax

```
Result &= Expression
```

Arguments

Result

Specify a variable for storing the result.

Expression

Write any expression.

Description

The &= operator performs string concatenation on the assumption that values of *Result* and *Expression* are all strings. If both values are Empty, an Empty value is assigned to *Result*. If *Result* is an Empty value or undefined, the value of *Expression* is assigned as is to *Result*. If *Expression* is an Empty value, *Result* remains unchanged.

Examples

```
' This code stores "ABCDEF" in variable result1.
result1 = "ABC"
result1 &= "DEF"

' This code stores "01" in variable result2.
result2 = 0
result2 &= 1

' This code stores "100" in variable result3.
Dim result3
result3 &= 100
```

JP1/Script version

Supported from JP1/Script 06-51.

8.3.16 AddStr (concatenate strings with delimiters inserted)

Purpose

Performs string concatenation on two or more strings and returns the concatenated string with specified delimiters inserted.

Syntax

```
AddStr ([SeparateChar], [Option], String1, String2 [,String3, ...])
```

Arguments

SeparateChar

Specify the delimiter as a character string or as a variable that stores this value.

This value is optional. If you omit this value, no delimiters are inserted.

Option

Specify either of the following values:

Value	Meaning
NeedDq	Enclose the result string with double quotation marks only if <i>String1</i> to <i>String5</i> include a comma, tab, or space taken as a delimiter.
AllDq	Enclose the result string with double quotation marks regardless of the strings specified in <i>String1</i> to <i>String5</i> .

This value is optional. If you omit this value, *NeedDq* is assumed.

String1 to *String5*

Write the strings to be concatenated. Specify each as a character string or as a variable that stores this value. You can write up to five strings.

Description

The *AddStr* command performs string concatenation on two or more strings and returns the concatenated string with specified delimiters inserted. The concatenated string is truncated to 1,024 characters if it exceeds this length.

Examples

```
' This command stores "'Code 291" Price "3,000"' in
' variable string1.
Dim string1
```

```
string1 = AddStr ("", , "Code 291", "Price", "3,000")

' This commands stores ""Code 291" "Price" "3,000"" in
' variable string2.
Dim string2
string2 = AddStr ("", AllDq, "Code 291", "Price",
                "3,000")
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.17 SeparateStrCount (count the number of separate strings)

Purpose

Splits a specified string at a specified delimiter, and returns the number of separate strings.

Syntax

```
SeparateStrCount (String, SeparateChar)
```

Arguments

String

Write the string to be split, or specify a variable that stores this value.

SeparateChar

Specify the delimiter as a character string or as a variable that stores this value.

Description

The `SeparateStrCount` command splits a specified string at the specified delimiter, and returns the number of separate strings. The command returns 1 if the delimiter is not found.

If *String* begins or ends with the delimiter, the command counts the separate strings assuming a zero-length string ("") immediately before or after that delimiter.

Zero is returned if *String* is a zero-length string ("").

To obtain the separated strings, use the `SeparateStr` command.

To include *n* double quotation marks (") as characters in *String*, you must enter $n*8$ double quotation marks. However, if you specify 2 in the following registry, you can use $n*2$ double quotation marks instead of $n*8$:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
ParseDQ
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify $n*2$ double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify $n*2$ double quotation marks (") is enabled.

To specify both 1 (see the `SetGV`, `GetGV`, and `DeleteGV` commands) and 2 as the value of this registry, type 3.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Examples

```
' This code stores 3 in variable count1.
Dim count1
count1 = SeparateStrCount ("JP1  Script  01-00,01-01", " ")

' This code stores 3 in variable count2.
Dim count2, param
param = ""Code 100"" Price 300"
count2 = SeparateStrCount (param, " ")

' This code stores 4 in variable count3.
Dim count3
count3 = SeparateStrCount ("Code100;Code200;Code300;", ";")
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.18 SeparateStr (split a string into separate strings)

Purpose

Splits a specified string at a specified delimiter, and returns the separated string.

Syntax

```
SeparateStr (String, SeparateChar [, Position])
```

Arguments

String

Write the string to be split, or specify a variable that stores this value.

SeparateChar

Specify the delimiter as a character string or as a variable that stores this value.

Position

Specify the position from which to split the string as the offset from beginning, where 1 is the first character. The delimiter is not included in the count from the first character.

A zero-length string ("") is returned if the *Position* value exceeds the number of characters in *String*.

This value is optional. If you omit this value, 1 is assumed.

Description

The `SeparateStr` command divides a specified string using the specified delimiter, and returns the string separated from the specified position.

If two or more delimiters occur one after the other in the specified string, a zero-length string ("") is returned as the separated string.

To obtain the number of separate strings, use the `SeparateStrCount` command.

To include n double quotation marks (") as characters in *String*, you must enter $n*8$ double quotation marks. However, if you specify 2 in the following registry, you can use $n*2$ double quotation marks instead of $n*8$:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX
```

Value name

```
ParseDQ
```

Value datatype

```
REG_DWORD
```

Value

0: If you are using quotation marks in a parameter string, the setting that allows you to specify $n*2$ double quotation marks (") is disabled (this is the default value).

1: If you are using quotation marks in a parameter string, the setting that allows you to specify $n*2$ double quotation marks (") is enabled.

To specify both 1 (see the `SetGV`, `GetGV`, and `DeleteGV` commands) and 2 as the value of this registry, type 3.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Examples

```
' This code stores "01-00,01-01" in variable string1.
Dim string1
string1 = SeparateStr ("JP1  Script  01-00,01-01", " ",3)

' This code stores a zero-length string ("" ) in variable
' string2.
Dim string2
string2 = SeparateStr ("JP1,Script,01-00,,01-01", ",", 4)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.3.19 Str (convert a number to a string)

Purpose

Returns a specified numeric value as a string.

Syntax

```
Str (Number)
```

Arguments

Number

Write the number to be converted or specify a variable that stores this value.

If you specify a string in *Number*, that string will be returned.

Description

The `Str` command returns the specified numeric value as a string.

Examples

```
' This code stores 2 in variable string1.
Dim string1
string1 = 1 + 2 - 1

' This code stores "12-1" in variable string2
Dim string2
string2 = Str (1) + Str (2) + Str (-1)

' This code stores 11 in variable string3.
Dim string3
string3 = Str (1) + 2 - 1
```

JP1/Script version

Supported from JP1/Script 05-20.

8.3.20 Format (convert a value to a formatted string)

Purpose

Returns a specified value as a formatted string.

Syntax

```
Format (Form, Arg1 [, Arg2, ...])
```

Arguments

Form

Specify the format into which *Arg1* to *Arg32* are to be converted. Write a character string or a variable that stores this value.

Each character specified in *Form* is represented as is, and the % symbol shows where the formatting specification starts.

Form can be any combination of the following:

Designation	Meaning
%d	Represent numbers as decimals.
%x	Represent numbers as hexadecimal (lowercase).
%X	Represent numbers as hexadecimal (uppercase).
%o	Represent numbers as octals.
%s	Represent strings as is.
%c	Represent a single character as it is. If any of the values in <i>Arg1</i> to <i>Arg32</i> consist of two or more characters, the first character is represented.
%5d, %10s, etc.	Specify the maximum number of digits to be formatted.
%05d	Pad the value with leading zeros if it has fewer than the specified maximum number of digits to be formatted.
%-5d, %-10s, etc.	Left-justify the value.

To enter % as the character itself, not as the formatting indicator, type %%.

Arg1 to Arg32

Specify each value to be formatted as a string, number, or as a variable that stores this value. You can specify multiple values in the same order as the format designations in *Form*.

Description

The `Format` command returns a specified value as a string.

Example

```
' Convert the value of global variable seqNo to a five-digit
' decimal number padded with leading zeros, and pass it to
' executable file ABC.EXE as a parameter.
' For example, if the value seqNo is "1", pass parameter
' "00001".
Dim numID, strID
numID = GetGV ("seqNo")
If IsEmpty (numID) Then
    numID = 1
End If
strID = Format ("%05d", numID)
Exec (_SCF_+"ABC.EXE", True, strID)
```

JP1/Script version

Supported from JP1/Script 05-20.

8.3.21 IsLower (check whether a string is lowercase)

Purpose

Checks whether a string is lowercase characters, and returns `True` or `False`.

Syntax

```
IsLower (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

`False` is returned if you specify a zero-length string (`""`).

Description

The `IsLower` command checks whether the specified string contains all lowercase characters, and returns `True` or `False` as the command execution result.

Example 1

```
' Check whether the value of global variable Level is
' "a" or "A".
Dim buff
buff = GetGV ("Level")
If buff = "A" Then
    If IsLower (buff) Then
        MsgBox ("Small A Level.")
    Else
        MsgBox ("Large A Level.")
    End If
End If
```

```
End If
End If
```

Example 2

```
' Check whether the string stored in local variable buff
' consists of all lowercase characters.
Dim buff
buff = "script"
If IsLower ( buff ) Then
    MessageBox ( "All characters are lowercase characters." )
Else
    MessageBox ( "Characters other than lowercase characters exist." )
End If
```

JP1/Script version

Supported from JP1/Script 06-00.

8.3.22 IsUpper (check whether a string is uppercase)

Purpose

Checks whether a string is uppercase characters, and returns `True` or `False`.

Syntax

```
IsUpper (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

`False` is returned if you specify a zero-length string (`""`).

Description

The `IsUpper` command checks whether the specified string contains all uppercase characters, and returns `True` or `False` as the command execution result.

Example 1

```
' Check whether the value of global variable Level is
' "A" or "a".
Dim buff
buff = GetGV ("Level")
If buff = "A" Then
    If IsUpper (buff) Then
        MessageBox ( "Level : Large A." )
    Else
        MessageBox ( "Level : Small A." )
    End If
End If
```

Example 2

```
' Check whether the string stored in local variable buff
' consists of all uppercase characters.
Dim buff
```

```

buff = "SCRIPT"
If IsUpper ( buff ) Then
    MessageBox ( "All characters are uppercase characters." )
Else
    MessageBox ( "Characters other than uppercase characters exist." )
End If

```

JP1/Script version

Supported from JP1/Script 06-00.

8.3.23 IsSingleChar (check whether a string is one-byte characters)

Purpose

Checks whether a string is one-byte characters, and returns True or False.

Syntax

```
IsSingleChar (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

False is returned if you specify a zero-length string ("").

Description

The IsSingleChar command checks whether the specified string contains one-byte characters only, and returns True or False as the command execution result.

Example 1

```

' Check the value of global variable seqNo, and display "#"
' before a one-byte character or "number" before a two-byte
' character.
Dim buff
buff = GetGV ( "seqNo" )
If IsSingleChar (buff) Then
    MessageBox ( "#" + buff)
Else
    MessageBox ( "item-number" + buff)
End If

```

Example 2

```

' Check whether the string stored in local variable buff
' consists of all one-byte characters.
Dim buff
buff = "JP1/Script"
If IsSingleChar ( buff ) Then
    MessageBox ( "All characters are one-byte characters." )
Else
    MessageBox ( "Characters other than one-byte characters exist." )
End If

```

JP1/Script version

Supported from JP1/Script 06-00.

8.3.24 IsMultiChar (check whether a string is two-byte characters)

Purpose

Checks whether a string is two-byte characters, and returns True or False.

Syntax

```
IsMultiChar (String)
```

Argument

String

Specify the string itself or a variable that stores this value.

False is returned if you specify a zero-length string ("").

Description

The IsMultiChar command checks whether the specified string contains two-byte characters only, and returns True or False as the command execution result.

Example 1

```
' Check the value of global variable seqNo, and display "#"
' before a two-byte character or "number" before a one-byte
' character.
Dim buff
buff = GetGV ( "seqNo" )
If IsMultiChar ( buff ) Then
    MsgBox ( "item-number" + buff)
Else
    MsgBox ( "#" + buff )
End If
```

Example 2

```
' Check whether the string stored in local variable buff
' consists of all two-byte characters.
Dim buff
buff = " JP1/SCRIPT "
If IsMultiChar ( buff ) Then
    MsgBox ( "All characters are two-byte characters." )
Else
    MsgBox ( "Characters other than two-byte characters exist." )
End If
```

JP1/Script version

Supported from JP1/Script 06-00.

8.4 Commands for working with dates

8.4.1 Date (return the current date)

Purpose

Returns the current date.

Syntax

`Date`

This command has no arguments.

Description

The `Date` command returns the current date in *yyyy/mm/dd* format.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.2 Time (return the current time)

Purpose

Returns the current time.

Syntax

`Time`

This command has no arguments.

Description

The `Time` command returns the current time in *hh:mm:ss* format.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.3 Year (return the year for a specified date)

Purpose

Returns a four-digit numeric representing the year for a specified date.

Syntax

```
Year ([Date])
```

Argument

Date

Write the date in *yyyy/mm/dd* format. Numbers specified in the range 0 to 69 in *yyyy* are read as 2000 to 2069. If an invalid value is specified for *mm*, an error occurs. However, if an invalid value is specified for *dd*, no error occurs. Instead, the `Year` command replaces the specified date with the equivalent valid date and returns the result. For example, if 1999/12/32 (a nonexistent date) were specified, the `Year` command would replace the specified date with the valid date 2000/01/01 and would return 2000.

This value is optional. If you omit this value, the current date is assumed.

Description

The `Year` command returns a four-digit numeric representing the year of the specified date.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.4 Month (return the month for a specified date)

Purpose

Returns a one- or two-digit numeric in the range 1 to 12, representing the month for a specified date.

Syntax

```
Month ([Date])
```

Argument

Date

Write the date in `yyyy/mm/dd` format. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069.

If an invalid value is specified for `mm`, an error occurs. However, if an invalid value is specified for `dd`, no error occurs. Instead, the `Month` command replaces the specified date with the equivalent valid date and returns the result. For example, if 1999/12/33 (a nonexistent date) were specified, the `Month` command would replace the specified date with the valid date 2000/01/02 and would return 1.

This value is optional. If you omit this value, the current date is assumed.

Description

The `Month` command returns a one- or two-digit numeric in the range 1 to 12, representing the month of the specified date.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.5 Day (return the day for a specified date)

Purpose

Returns a one- or two-digit numeric in the range 1 to 31, representing the day of the specified date.

Syntax

```
Day ([Date])
```

Argument

Date

Write the date in `yyyy/mm/dd` format. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069.

If an invalid value is specified for `mm`, an error occurs. However, if an invalid value is specified for `dd`, no error occurs. Instead, the `Day` command replaces the specified date with the equivalent valid date and returns the result. For example, if 1999/12/33 (a nonexistent date) were specified, the `Day` command would replace the specified date with the valid date 2000/01/02 and would return 2.

This value is optional. If you omit this value, the current date is assumed.

Description

The `Day` command returns a one- or two-digit numeric in the range 1 to 31, representing the day of the specified date.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.6 Weekday (return the weekday for a specified date)

Purpose

Returns a one-digit number in the range 1 (Sunday) to 7 (Saturday), representing the day of the week for a specified date.

Syntax

```
Weekday ([Date] [, Option])
```

Arguments

Date

Write the date in `yyyy/mm/dd` format. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069. If an invalid value is specified for `mm`, an error occurs. However, if an invalid value is specified for `dd`, no error occurs. Instead, the `Weekday` command replaces the specified date with the equivalent valid date and returns the result. For example, if `1999/12/33` (a nonexistent date) were specified, the `Weekday` command would replace the specified date with the valid date `2000/01/02` and would return 1 (for Sunday).

This value is optional. If you omit this value, the current date is assumed.

Option

Specify one of the following values:

Value	Meaning
String	Return an abbreviation as the day of the week.
StringJ	Return the full name as the day of the week.

Description

If you omit the *Option* argument, the `Weekday` command returns a single-digit number in the range 1 (Sunday) to 7 (Saturday), as follows:

Value	Meaning
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

If you specify the *Option* argument, the return value is one of the following strings:

String specified in Option	StringJ specified in Option
"SUN"	"Sunday"
"MON"	"Monday"
"TUE"	"Tuesday"
"WED"	"Wednesday"
"THU"	"Thursday"
"FRI"	"Friday"
"SAT"	"Saturday"

JP1/Script version

Supported from JP1/Script 01-00.

8.4.7 Hour (return the hour for a specified time)

Purpose

Returns a one- or two-digit numeric in the range 0 to 23, representing the hour of the specified time.

Syntax

```
Hour ([Time])
```

Argument

Time

Write the time in *hh:mm:ss* format.

If an invalid value is specified for *hh*, an error occurs. However, if an invalid value is specified for *mm* or *ss*, no error occurs.

For example, if 19:61:00 (a nonexistent time) were specified, the Hour command would replace the specified time with the valid time 20:01:00 and would return 20.

This value is optional. If you omit this value, the current time is assumed.

Description

The Hour command returns a one- or two-digit numeric in the range 0 to 23, representing the hour of the specified time.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.8 Minute (return the minute for a specified time)

Purpose

Returns a one- or two-digit numeric in the range 0 to 59, representing the minute of the specified time.

Syntax

```
Minute ([Time])
```

Argument

Time

Write the time in *hh:mm:ss* format.

If an invalid value is specified for *hh*, an error occurs. However, if an invalid value is specified for *mm* or *ss*, no error occurs. Instead, the `Minute` command replaces the specified time with the equivalent valid time and returns the result.

For example, if `19:61:00` (a nonexistent time) were specified, the `Minute` command would replace the specified time with the valid time `20:01:00` and would return 1.

This value is optional. If you omit this value, the current time is assumed.

Description

The `Minute` command returns a one- or two-digit numeric in the range 0 to 59, representing the minute of the specified time.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.9 Second (return the second for a specified time)

Purpose

Returns a one- or two-digit numeric in the range 0 to 59, representing the second of the specified time.

Syntax

```
Second ([Time])
```

Argument

Time

Write the time in *hh:mm:ss* format.

If an invalid value is specified for *hh*, an error occurs. However, if an invalid value is specified for *mm* or *ss*, no error occurs. Instead, the `Second` command replaces the specified time with the equivalent valid time and returns the result.

For example, if `19:00:65` (a nonexistent time) were specified, the `Second` command would replace the specified time with the valid time `19:01:05` and would return 5.

This value is optional. If you omit this value, the current time is assumed.

Description

The `Second` command returns a one- or two-digit numeric in the range 0 to 59, representing the second of the specified time.

JP1/Script version

Supported from JP1/Script 01-00.

8.4.10 CalcDate (add and subtract dates)

Purpose

Adds or subtracts a specified number of years, months, or days to or from a specified date, and returns the calculation result.

Syntax

```
CalcDate (Date, Calc, [Years], [Months] [, Days])
```

Arguments

Date

Write the date in *yyyy/mm/dd* format. Numbers specified in the range 0 to 69 in *yyyy* are read as 2000 to 2069. If an invalid value is specified for *mm*, an error occurs. However, if an invalid value is specified for *dd*, no error occurs. Instead, the `CalcDate` command replaces the specified date with the equivalent valid date and returns the result.

For example, if 1999/12/33 (a nonexistent date) were specified, the `CalcDate` command would replace the specified date with the valid date 2000/01/02.

Calc

Specify the calculation to perform. Write either of the following values:

Value	Meaning
Minus	Subtraction: Return the date specified in <i>Date</i> , minus the years, months, or days specified in <i>Years</i> , <i>Months</i> , and/or <i>Days</i> .
Plus	Addition: Return the date specified in <i>Date</i> , plus the years, months, or days specified in <i>Years</i> , <i>Months</i> , and/or <i>Days</i> .

Years

Write the years as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

Months

Write the months as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

Days

Write the days as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

Description

The `CalcDate` command adds or subtracts a specified number of years, months, or days to or from a specified date, and returns the calculation result in *yyyy/mm/dd* format.

If the calculated date does not exist, the command returns the closest future date. For example, if the calculated date is 1999/02/29 (non-existent), the returned date will be 1999/03/01 (existent).

A zero-length string ("") is returned if an error occurs.

Example

```
' Display the date 10 days before today.
Dim result1
result1 = CalcDate (Date(), Minus, , , 10)
MessageBox ("The date 10 days ago is "+result1+".")
```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.11 CompDate (compare dates)

Purpose

Compares two specified dates and returns `True` or `False`.

Syntax

```
CompDate (Date1, Comp, Date2)
```

Arguments

Date1

Write one of the dates to be compared in `yyyy/mm/dd` format. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069.

If an invalid value is specified for `mm`, an error occurs. However, if an invalid value is specified for `dd`, no error occurs. Instead, the `CompDate` command replaces the specified date with the equivalent valid date and returns the result.

For example, if 1999/12/33 (a nonexistent date) were specified, the `CompDate` command would replace the specified date with the valid date 2000/01/02.

Comp

Specify how to compare the dates. Write one of the following values:

Value	Meaning
Equal	Equal (=)
NotEqual	Not equal (<>)
Before	<i>Date2</i> comes before <i>Date1</i> (>)
After	<i>Date2</i> comes after <i>Date1</i> (<)

Date2

Write the other date in `yyyy/mm/dd` format. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069. If an invalid value is specified for `mm`, an error occurs. However, if an invalid value is specified for `dd`, the `CompDate` command replaces the specified date with the equivalent valid date.

Description

The `CompDate` command compares two dates using the method specified in `Comp`, and returns `True` or `False` as the command execution result.

A zero-length string ("") is returned if an error occurs.

Example

```
' Delete the files in the temporary folder that
' were created 30 or more days ago.
Dim fileName, delDate, creDate
delDate = CalcDate (Date(), Minus, 0, 0, 29)

For fileName = _TEMP_+"*.*" Do
  GetFileTime (_TEMP_+fileName, creDate, , Create)
  If CompDate (creDate, After, delDate) = True Then
    DeleteFile (_TEMP_+fileName)
  End If
End For
```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.12 GetDateCount (calculate the difference between dates)

Purpose

Calculates the difference between two specified dates.

Syntax

```
GetDateCount (StartDate, EndDate [, UnitofDate])
```

Arguments

StartDate

Write the date from which to start the count in *yyyy/mm/dd* format. Numbers specified in the range 0 to 69 in *yyyy* are read as 2000 to 2069.

If an invalid value is specified for *mm*, an error occurs. However, if an invalid value is specified for *dd*, no error occurs. Instead, the `GetDateCount` command replaces the specified date with the equivalent valid date and returns the result.

For example, if 1999/12/33 (a nonexistent date) were specified, the `GetDateCount` command would replace the specified date with the valid date 2000/01/02.

EndDate

Write the date at which to end the count in *yyyy/mm/dd* format. Numbers specified in the range 0 to 69 in *yyyy* are read as 2000 to 2069. If an invalid value is specified for *mm*, an error occurs. However, if an invalid value is specified for *dd*, the `GetDateCount` command replaces the specified date with the equivalent valid date.

UnitofDate

Specify the units in which to calculate the elapsed time as one of the following values:

Value	Meaning
YearU	Get the number of elapsed years. For example, if the start date is 1998/4/1 and the end date is 1999/4/1, the calculation result will be 1. If the end date is 1999/3/1, the calculation result will be 0.
MonthU	Get the number of elapsed months. For example, if the start date is 1998/4/10 and the end date is 1998/5/10, the calculation result will be 1. If the end date is 1998/5/9, the calculation result will be 0.
DayU	Get the number of elapsed days.

This value is optional. If you omit this value, `DayU` is assumed.

Description

The `GetDateCount` command calculates the difference between two specified dates, and returns the elapsed time as the command execution result. If the date specified in *StartDate* is later than the date specified in *EndDate*, a negative value is returned.

Example

```
' Calculate the elapsed time from 1998/4/1 to 1999/3/31  
' in years, months, and days.  
Dim date1, date2, yBuff, mBuff, dBuff  
date1 = "1998/4/1"
```

```

date2 = "1999/3/31"

' The following code stores 0 in yBuff, 11 in mBuff,
' and 364 in dBuff.
yBuff = GetDateCount (date1, date2, YearU)
mBuff = GetDateCount (date1, date2, MonthU)
dBuff = GetDateCount (date1, date2, DayU)

```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.13 CalcTime (add and subtract times)

Purpose

Adds or subtracts a specified number of hours, minutes, or seconds to or from a specified time, and returns the calculation result.

Syntax

```
CalcTime (Time, Calc, [Hours], [Minutes], [Seconds] [, DaysBuff])
```

Arguments

Time

Write the time in *hh:mm:ss* format.

If an invalid value is specified for *hh*, an error occurs. However, if an invalid value is specified for *mm* or *ss*, no error occurs. Instead, the `CalcTime` command replaces the specified time with the equivalent valid time.

For example, if 19:00:65 (a nonexistent time) were specified, the `CalcTime` command would replace the specified time with the valid time 19:01:05.

Calc

Specify the calculation to perform. Write either of the following values:

Value	Meaning
Minus	Subtraction: Return the time specified in <i>Time</i> , minus the hours, minutes, or seconds specified in <i>Hours</i> , <i>Minutes</i> , and/or <i>Seconds</i> .
Plus	Addition: Return the time specified in <i>Time</i> , plus the hours, minutes, or seconds specified in <i>Hours</i> , <i>Minutes</i> , and/or <i>Seconds</i> .

Hours

Write the hours as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

Minutes

Write the minutes as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

Seconds

Write the seconds as 0 or a number, or as a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

DaysBuff

Specify a variable for storing the number of days when the calculated time falls on a different day. Omit this value if not required.

Zero is set in this variable if the result is within the same day.

Description

The `CalcTime` command adds or subtracts a specified number of hours, minutes, or seconds to or from a specified time, and returns the calculation result in `hh:mm:ss` format.

A zero-length string ("") is returned if an error occurs.

Example

```
' Queue a script file to execute automatically 7 hours
' and 45 minutes after the current time.
Dim resTime, resdate, daysBuff
resTime = CalcTime (Time(), Plus, 7, 45, , daysBuff)
resDate = CalcDate (Date(), Plus, , , daysBuff)
EntryStartUp (_SCF_"ABC.SPT", , , resTime, , Day(resDate))
```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.14 CompTime (compare times)

Purpose

Compares two specified times and returns `True` or `False`.

Syntax

```
CompTime (Time1, Comp, Time2)
```

Arguments

Time1

Write one of the times to be compared in `hh:mm:ss` format.

If an invalid value is specified for `hh`, an error occurs. However, if an invalid value is specified for `mm` or `ss`, no error occurs. Instead, the `CompTime` command replaces the specified time with the equivalent valid time.

For example, if `19:00:65` (a nonexistent time) were specified, the `CompTime` command would replace the specified time with the valid time `19:01:05`.

Comp

Specify how to compare the times. Write one of the following values:

Value	Meaning
Equal	Equal (=)
NotEqual	Not equal (<>)
Before	<i>Time2</i> comes before <i>Time1</i> (>)
After	<i>Time2</i> comes after <i>Time1</i> (<)

Time2

Write the other time in `hh:mm:ss` format.

If an invalid value is specified for `mm` or `ss`, the `CompTime` command replaces the specified time with the equivalent valid time.

Description

The `CompTime` command compares two times using the method specified in *Comp*, and returns `True` or `False` as the command execution result.

A zero-length string ("") is returned if an error occurs.

Example

```
' Delete files created this morning.
Dim fileName, creDate, creTime

For fileName = _TEMP_+"*.*" Do
  GetFileTime (_TEMP_+fileName, creDate, creTime, Create)
  If CompDate (creDate, Equal, Date()) Then
    If CompTime (creTime, After, "12:00:00") = True Then
      DeleteFile (_TEMP_+fileName)
    End If
  End If
End For
```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.15 GetTimeCount (calculate the difference between times)

Purpose

Calculates the difference between two specified times.

Syntax

```
GetTimeCount (StartTime, EndTime [, UnitofTime])
```

Arguments

StartTime

Write the time from which to start the count in *hh:mm:ss* format.

If an invalid value is specified for *hh*, an error occurs. However, if an invalid value is specified for *mm* or *ss*, no error occurs. Instead, the `GetTimeCount` command replaces the specified time with the equivalent valid time.

For example, if 19:00:65 (a nonexistent time) were specified, the `GetTimeCount` command would replace the specified time with the valid time 19:01:05.

EndTime

Write the time at which to end the count in *hh:mm:ss* format.

If an invalid value is specified for *mm* or *ss*, the `GetTimeCount` command replaces the specified time with the equivalent valid time.

UnitofTime

Specify the units in which to calculate the elapsed time as one of the following values:

Value	Meaning
HourU	Get the number of elapsed hours. For example, if the start time is 9:00:00 and the end time is 10:00:00, the calculation result will be 1. If the end time is 9:59:59, the calculation result will be 0.
MinuteU	Get the number of elapsed minutes.

Value	Meaning
MinuteU	For example, if the start time is 9:10:00 and the end time is 9:11:00, the calculation result will be 1. If the end time is 9:10:59, the calculation result will be 0.
SecondU	Get the number of elapsed seconds.

This value is optional. If you omit this value, SecondU is assumed.

Description

The `GetTimeCount` command calculates the difference between two specified times, and returns the elapsed time as the command execution result. If the time specified in *StartTime* is later than the time specified in *EndTime*, a negative value is returned.

Example

```
' Calculate the elapsed time from 9:10:30 to 10:09:20
' in hours, minutes, and seconds.
Dim time1, time2, hBuff, mBuff, sBuff
time1 = "9:10:30"
time2 = "10:09:20"

' The following code stores 0 in hBuff, 58 in mBuff,
' and 3530 in sBuff.
hBuff = GetTimeCount (time1, time2, HourU)
mBuff = GetTimeCount (time1, time2, MinuteU)
sBuff = GetTimeCount (time1, time2, SecondU)
```

JP1/Script version

Supported from JP1/Script 05-20.

8.4.16 IsLeapYear (check whether a leap year)

Purpose

Checks whether a specified year is a leap year and returns True or False.

Syntax

```
IsLeapYear (Year)
```

Arguments

Year

Write the year as a number or as a variable that stores this value. Numbers specified in the range 0 to 69 in *Year* are read as 2000 to 2069.

Description

The `IsLeapYear` command checks whether the specified year is a leap year and returns True or False.

Example

```
' Check whether this year is a leap year.
Dim nowYear
nowYear = Year (Date())
If IsLeapYear (nowYear) Then
    MessageBox ("This year is a leap year.")
Else
```

```
    MsgBox ("This year is not a leap year.")  
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

8.5 Commands for managing files and folders

8.5.1 IniRead (read a value from an initialization file)

Purpose

Reads a value from an initialization file (.INI file).

Syntax

```
IniRead (FilePath, SectionName, ValueBuff, EntryName)
```

Arguments

FilePath

Write the full path of the initialization file as a character string or as a variable that stores this value.

SectionName

Write a section name as a character string or as a variable that stores this value.

ValueBuff

Specify a variable for storing the read data.

EntryName

Write an entry name as a string, number, or as a variable that stores this value.

Description

The `IniRead` command reads the value set in an entry of a specific section of the specified initialization file (.INI file) and stores the value in a specified variable. The command returns `True` on successful execution, or `False` if an error occurs.

If the initialization file specified in *FilePath* or the specific section of the initialization file specified in *SectionName* does not exist, a zero-length string ("") is stored in the variable specified in *ValueBuff*. Then the command returns `True` as the execution result.

Note

Take care when specifying an initialization file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Important note

If the value in an entry contains more than 1,024 bytes, only the first 1,024 bytes are stored in a variable. Do not attempt to read an entry containing more than 1,024 bytes.

Example

The script below reads the following data from the initialization file `ABC.INI` in the startup folder.

```
; ABC.INI
[Files]
File01=Readme.txt
File02=Abc.exe

' Script file
' Line 1 stores "Readme.txt" in variable file1.
' Line 2 stores "Abc.exe" in variable file2.
```

```
Dim file1, file2
IniRead (_BIN_"ABC.INI", "Files", file1, "File01")
IniRead (_BIN_"ABC.INI", "Files", file2, "File02")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.2 IniWrite (enter a value in an initialization file)

Purpose

Enters a value in an initialization file (.INI file).

Syntax

```
IniWrite (FilePath, SectionName, EntryName, Value)
```

Arguments

FilePath

Write the full path of the initialization file as a character string or as a variable that stores this value.

SectionName

Write a section name as a character string or as a variable that stores this value.

EntryName

Write an entry name as a string, number, or as a variable that stores this value.

Value

Specify the value to set in this entry as a string, number, or as a variable that stores this value.

Description

The `IniWrite` command enters a value in a specific section of the specified initialization file. The command returns `True` on successful execution, or `False` if an error occurs.

If the specified initialization file does not exist, a new file is created. If the specified section does not exist, a new section is created.

Notes

- Take care when specifying an initialization file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating an initialization file. In the following registry key, set the access permissions beforehand:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option
```

Value name

```
SecurityAttributesSucceed
```

Value datatype

```
REG_DWORD
```

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

The script below enters data in the initialization file ABC.INI in the Windows folder.

```
' Script file
IniWrite (_WIN_"ABC.INI", "Files", "File01", "Readme.txt")
IniWrite (_WIN_"ABC.INI", "Files", "File02", "Abc.exe")

; ABC.INI
[Files]
File01 = Readme.txt
File02 = Abc.exe
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.3 TextFileReplace (replace a string in a text file)

Purpose

Replaces a specified string in a text file.

Syntax

```
TextFileReplace (FileName, OldText, NewText [, ReplaceCntBuff])
```

Arguments

FileName

Specify a text file name as a character string or as a variable that stores this value.

If you omit the extension, .TXT is automatically appended to the file name. However, in JP1/Script 06-00 and later versions, if you specify a file name ending with the extension period (.), the file is regarded as having no extension.

OldText

Write the string to be replaced, or specify a variable that stores this value.

NewText

Write the new string, or specify a variable that stores this value.

ReplaceCntBuff (from version 06-00)

Specify a variable for storing the number of strings replaced by the command. Omit this argument if not required.

Description

The TextFileReplace command searches for the specified string in the text file and replaces it with the new string. The command returns True on successful execution, or False if an error occurs.

This command does not support large files. If necessary, use the command from SplitFile (split a file).

Note

Take care when specifying a file in the folder set in the environment variable ProgramFiles (normally the Program Files folder on the system drive) or WinDir (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Important note

The command does not convert a linefeed code in the file to the new character string even if the linefeed code is specified as the old character string.

Example

The script below executes the `TextFileReplace` command on the file `ABC.TXT`.

```
TextFileReplace ( _BIN_+"ABC.TXT" , "TEST" , "test" )
```

[Contents of the `ABC.TXT` file before execution]

```
*****  
**TEST**  
*****
```

[Contents of the `ABC.TXT` file after execution]

```
*****  
**test**  
*****
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.4 TextOpen (open a text file)

Purpose

Opens a text file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
TextOpen (FilePath [, Mode])
```

Arguments

FilePath

Write the full path of the text file as a character string or as a variable that stores this value.

If you omit the extension, `.TXT` is automatically appended to the file name. However, in JP1/Script 06-00 and later versions, if you specify a file name ending with the extension period (`.`), the file is regarded as having no extension.

Mode

Specify how to open the file as one of the following values:

Value	Meaning
Create	Create a new file unconditionally.

Value	Meaning
ReadOnly	Open the file in read-only mode.
WriteOnly	Open the file in write-only mode.
ReadWrite	Open the file in read/write mode.

If you specify `ReadOnly` in *Mode*, the file will be accessed in shared mode. If you specify `Create`, `WriteOnly`, or `ReadWrite`, the file will be locked when accessed.

This value is optional. If you omit this value, `Create` is assumed.

Description

The `TextOpen` command opens a specified text file. The command returns the file ID on successful execution, or 0 if an error occurs.

At successful execution, the current read/write position is the beginning of the file (0).

Notes

- Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating a text file. In the following registry key, set the access permissions beforehand:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option

Value name

SecurityAttributesSucceed

Value datatype

REG_DWORD

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

```
' Read Logging.txt, and display its contents in a dialog box.
Dim file1
file1 = TextOpen (_BIN_+"Logging.txt", ReadOnly)
If file1 = 0 Then
    MsgBox (_BIN_+"Failed to open Logging.txt", OK)
Else
    Dim buff1
    If TextRead (file1, buff1) Then
        MsgBox (buff1, OK)
    End
End

TextClose (file1)
End
```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.5 TextClose (close a text file)

Purpose

Closes a text file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
TextClose ([FileId])
```

Argument

FileId

Specify the file ID as a number or as a variable that stores this value.

This file ID is the execution result returned by the `TextOpen` command.

This value is optional. When the argument is omitted or is 0, the command closes all open files.

Description

The `TextClose` command closes the text file specified by file ID. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
' Read Logging.txt, and write its contents to Backup.txt.
Dim file1, file2
file1 = TextOpen (_BIN_+"Logging.txt", ReadOnly)
If file1 = 0 Then
    MessageBox (_BIN_+"Failed to open Logging.txt", OK)
    Exit
End
file2 = TextOpen (_TEMP_+"Backup.txt", Create)
If file2 = 0 Then
    MessageBox ("Unable to open" _TEMP_+"Backup.txt.", OK)
    TextClose (file1)
    Exit
End

Dim buff1
If TextRead (file1, buff1) Then
    TextWrite (file2, buff1)
End
TextClose (file1)
TextClose (file2)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.6 TextRead (read one line of data from a text file)

Purpose

Reads one line of data from a text file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
TextRead (FileId, Buff)
```

Arguments

FileId

Specify the file ID as a number or as a variable that stores this value.

This file ID is the execution result returned by the `TextOpen` command. Specify a value other than `WriteOnly` as the file access mode in the `TextOpen` command.

Buff

Specify a variable for storing the line of data read from the file. The data is truncated if it exceeds 1,024 bytes.

Description

The `TextRead` command reads one line of data from the text file specified by file ID, starting at the current read/write position, and stores the data in the specified variable. The command returns `True` on successful execution, or `False` if an error occurs.

On successful execution, the next line becomes the current read/write position.

If the end of the file is reached, the command returns `False` as the execution result and sets the same value as the `_ERR_EOF_` reserved variable in the `_RTN_` reserved variable. The execution control does not jump to the label specified in the `On Error` statement.

Important note

When the `TextRead` command detects the end of file, JP1/Script treats it as an error in the same manner as an error during command execution. In such a case, the error is recorded in the execution trace log even if script processing terminates normally. On the Trace Viewer, this event is displayed as an abnormal termination. Ignore any EOF detection error on the `TextRead` command.

Additionally, the variable for receiving a line of data read by the `TextRead` command contains the previous value. Therefore, after detection of EOF, do not use any process that expects the variable to contain no value.

Example

```
' The following script reads lines of data one by one from
' the Logging.txt text file in the execution folder.
Dim file1, buff1, readRtn

On Error GoTo ErrorBranch
file1 = TextOpen (_BIN_+"Logging.txt", ReadOnly)
TextRead (file1, buff1)
readRtn = _RTN_
While readRtn <> _ERR_EOF_
    MsgBox (buff1)
    TextRead (file1, buff1)
    readRtn = _RTN_
End
```

```
TextClose (file1)
Exit (0)

ErrorBranch:
  MessageBox ( _BIN_+"Operation on the Logging.txt file failed", OK)
  TextClose (file1)
  Exit (-1)
```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.7 TextWrite (write data to a text file)

Purpose

Writes data to a text file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
TextWrite (FileId, Data, [NewLine] [, Position])
```

Arguments

FileId

Specify the file ID as a number or as a variable that stores this value.

This file ID is the execution result returned by the `TextOpen` command. Specify a value other than `ReadOnly` as the file access mode in the `TextOpen` command.

Data

Specify the write data as a character string or as a variable that stores this value.

NewLine

Set `True` to start a new line after the data is written, or `False` to continue on the same line.

This value is optional. If you omit this value, `True` is assumed.

Position

Specify the read/write position relative to the beginning of the file (0). Write a number or a variable that stores this value. You can specify a value in the range from 0 to 2,147,483,647 (units: bytes).

This value is optional. If you omit this value, the current read/write position is assumed.

For large files, you cannot specify 2 gigabytes or a larger value as the write position.

Description

The `TextWrite` command writes data to the text file specified by file ID, starting at the current read/write position or at a specified read/write position. The command returns `True` on successful execution, or `False` if an error occurs.

On successful execution, the current read/write position is the next line if you specified `True` in *NewLine*, or the end of the written data if you specified `False`.

Important note

If you specify `WriteOnly` or `ReadWrite` in the *Mode* of the `TextOpen` command for an existing file and then execute the `TextWrite` command, the `TextWrite` command overwrites as many characters

as are specified. If the number of characters to be output (including linefeed codes) is less than the number in the existing file, all the excess characters contained in the file before the file was opened remain in the file.

Example

The script below creates the text file `Loging.txt` in the executable folder.

- Script file

```
Dim file1
file1 = TextOpen ( _BIN_"Loging.txt" ,Create )
If file1 = 0 Then
    MsgBox ( _BIN_"Failed to open Loging.txt" ,OK )
Else
    TextWrite ( file1 , "<< " + _COMP_ + " / " + _USER_ + " >>" )
    TextWrite ( file1 , Date + " " + Time + " " , False )
    TextWrite ( file1 , "Start script execution" )
    TextClose ( file1 )
End
```

- Execution result (contents of `Loging.txt`)

```
<< COMP01 / USER01 >>
2011/08/31 11:57:22 Start script execution
```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.8 TextSeek (move the read/write position to the file beginning or end)

Purpose

Moves the read/write position in a text file to the beginning or end of the file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
TextSeek ( FileId [, Point] )
```

Arguments

FileId

Specify the file ID as a number or as a variable that stores this value.

This file ID is the execution result returned by the `TextOpen` command. Specify a value other than `WriteOnly` as the file access mode in the `TextOpen` command.

Point

Specify the read/write position as either of the following values:

Value	Meaning
ToBegin	Move to the beginning of the file.
ToEnd	Move to the end of the file.

This value is optional. If you omit this value, `ToBegin` is assumed.

Description

The `TextSeek` command moves the read/write position to the specified position in the text file specified by file ID. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
' Write "*** END OF FILE ***" at the end of the text file
' "Loging.txt" in the execution folder.
Dim file1
file1 = TextOpen ( _BIN_"Loging.txt", ReadWrite)
If file1 = 0 Then
    MessageBox ( _BIN_"Failed to open Loging.txt", OK)
Else
    If TextSeek (file1, ToEnd) Then
        TextWrite (file1, "*** END OF FILE ***", False)
    End
    TextClose (file1)
End
```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.9 GetTextPosition (return the current read/write position)

Purpose

Returns the current read/write position in a text file.

Syntax

```
GetTextPosition (FileId)
```

Argument

FileId

Specify the file ID as a number or as a variable that stores this value.

This file ID is the execution result returned by the `TextOpen` command.

Description

The `GetTextPosition` command acquires the current read/write position in the text file specified by the file ID, as the number of bytes from the beginning of the file (0). The command returns the current read/write position on successful execution, or a zero-length string ("") if an error occurs.

An error results if the read/write start position is beyond 2,147,483,647.

Example

```
' Change the string "Start" in the text file "Loging.txt"
' in the execution folder to "Start script execution".
Dim file1
file1 = TextOpen ( _BIN_"Loging.txt" ,ReadWrite )
If file1 = 0 Then
    MessageBox ( _BIN_"Failed to open Loging.txt" ,OK )
Else
    Dim line ,position ,buff
    For line = 1 To 10
        ' Acquire the read/write position.
        position = GetTextPosition ( file1 )
```

```

If TextRead ( file1 ,buff ) Then
  If buff = "Start" Then
    ' Overwrite the text at the read/write position.
    TextWrite ( file1 ,"Start script execution" ,True ,position )
  End
Else
  Exit For
End
Next
TextClose (file1)
End

```

JP1/Script version

Supported from JP1/Script 05-00.

8.5.10 MakeDir (create a folder)

Purpose

Creates a folder.

Syntax

```
MakeDir (DirPath)
```

Argument

DirPath

Write the new folder path as a character string or as a variable that stores this value.

Description

The `MakeDir` command creates the specified folder. The command returns `True` on successful execution, or `False` if an error occurs.

If the path to the specified folder does not exist, the required path and folder are created.

Notes

- Take care when specifying a folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions for a folder you created. If the path to the specified folder does not exist, access permissions are also applied to creation of the path to the folder. In the following registry key, set the access permissions beforehand.

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option
```

Value name

```
SecurityAttributesSucceed
```

Value datatype

```
REG_DWORD
```

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

```
' Create a folder "HITACHI" in the Windows folder.
MakeDir (_WIN_"HITACHI")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.11 DeleteDir (delete a folder)

Purpose

Deletes a folder.

Syntax

```
DeleteDir (DirPath [, Option])
```

Arguments

DirPath

Write the folder path to be deleted, using a character string or a variable that stores this value.

Option (from version 06-00)

Specify either of the following values:

Value	Meaning
<i>Anyway</i>	Delete the entire folder even if it contains files.
<i>IfEmpty</i>	Delete the folder only if it is empty. <code>False</code> is returned as the command execution result if the folder is not deleted.

This value is optional. If you omit this value, *Anyway* is assumed.

Description

The `DeleteDir` command deletes the specified folder. The command returns `True` on successful execution, or `False` if an error occurs. `True` is also returned if you specify a non-existent folder.

Note

Take care when specifying a folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Important note

Take care in using this command. If you do not specify `If Empty`, the entire folder will be deleted even if it contains files.

Examples

```
' Unconditionally delete the folder "HITACHI" in the
' Windows folder.
```



```
DeleteDir (_WIN_"HITACHI")

' Delete the folder "HITACHI" in the Windows folder if
' it is empty.
DeleteDir (_WIN_"HITACHI", IfEmpty)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.12 DeleteFile (delete a file)

Purpose

Deletes a file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
DeleteFile (PathName [, Option])
```

Arguments

PathName

Write the full path of the folder or file to be deleted, using a character string or a variable that stores this value. You can use a wildcard in the file name.

Option (from version 05-10)

Specify the following optional value:

Value	Meaning
ExclDir	Delete only the files in the folder specified in <i>PathName</i> , not the folder itself.

Description

The `DeleteFile` command deletes the specified folder or file. If you omit the option, the command deletes the entire folder, even if it contains files. If you specify `ExclDir` in *Option*, only the files in the folder are deleted.

The command returns `True` on successful execution, or `False` if an error occurs. `True` is also returned if no such file can be found (same value as the `_ERR_FILE_` reserved variable) or if no such network name or path can be found (same value as the `_ERR_PATH_` reserved variable).

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Important note

The following types of file names are used in a Windows file system:

- Long file names with character strings specified by a user
- Short file names automatically generated by Windows (in 8.3 format)

Therefore, if a wildcard is specified in a file name in JP1/Script, the short file names automatically generated by the OS are also included. In addition, if a wildcard is not specified for a file name, the short file name

of another file that exists in the same path as the specified file might be the same, resulting in operations being performed on files you did not intend.

Examples

```
' Delete the "tempfile.tmp" file in the temporary folder.
DeleteFile (_TEMP_"tempfile.tmp")

' Delete all files with the extension ".tmp" in the
' temporary folder.
DeleteFile (_TEMP_"*.tmp")

' Delete all files in the temporary folder, but keep the
' folder itself.
DeleteFile (_TEMP_, ExclDir)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.13 Rename (rename a file)

Purpose

Renames a file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
Rename (OldName, [NewName] [, Method])
```

Arguments

OldName

Specify the old file name or folder name as a character string or as a variable that stores this value.

NewName

Specify the new file name or folder name as a character string or as a variable that stores this value. If you use a full path to specify the old file name or folder name, you must also specify a full path for the new file name or folder name.

If you omit this argument and specify `Reboot` as the update method, the file will be deleted.

Method

Specify the update method as one of the following values:

Value	Meaning
Replace	Rename the file even if a file of the name specified in <i>NewName</i> already exists.
NoReplace (from version 06-00)	Return an error occurs if a file of the name specified in <i>NewName</i> already exists.
Reboot	Rename the file at system restart.
FreeExt	Replace the file extension with any one of extensions .000 to .999 that is not already in use. If you specify this option, any new name is ignored. Unlike the <code>File</code> option of the <code>Copy</code> command, if an existing file with an extension from .000 to .999 has the same contents as the file specified in <i>NewName</i> , the <code>Rename</code> command simply deletes the file specified in <i>OldName</i> .

This value is optional. If you omit this value, `Replace` is assumed.

If you specify `Reboot`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Description

The `Rename` command renames the file specified in `OldName` as the file name specified in `NewName`. The command returns `True` on successful execution, or `False` if an error occurs.

If you omit or set a zero-length string ("") in `NewName`, and specify `Reboot` as the update method, the file will be deleted.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Examples

```
' Rename a file.
Dim outDir1
outDir1 = _BIN_
Rename (outDir1+"DEFAULT.DAT", outDir1+"DEFAULT.000",
      FreeExt)
Rename (TEMP_+"WORK.DAT", outDir1+"DEFAULT.DAT")

' Copy a file and delete the original (not otherwise
' possible).
Copy (_BIN_+"ABC.EXE", _BIN_+"XYZ.EXE", Overwrite)
Rename (_BIN_+"ABC.EXE", "", Reboot)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.14 TempDir (get the temporary folder name)

Purpose

Gets the temporary folder name.

Syntax

```
TempDir (DirNameBuff)
```

Argument

DirNameBuff

Specify a variable for storing the temporary folder name.

Description

The `TempDir` command gets the name of the system's temporary folder and stores it in a variable. The command returns `True` on successful execution, or `False` if an error occurs.

A backslash (\) is appended to the folder name.

JP1/Script version

Supported from JP1/Script 01-00.

8.5.15 TempFile (create a temporary file)

Purpose

Creates a temporary file.

Syntax

```
TempFile (FileNameBuff, [Prefix] [, DirName])
```

Arguments

FileNameBuff

Specify a variable for storing the created temporary file name.

Prefix

Specify the file prefix as a character string or as a variable that stores this value. The first three characters are valid.

This value is optional. If you omit this value, "STX" is assumed.

DirName

Specify the name of the folder in which to create the temporary file, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the system's temporary folder is assumed.

Description

The `TempFile` command creates a temporary file with a name in *prehexadecimal-string*. TMP format in the folder specified in *DirName*.

pre: The character string specified in *Prefix*

hexadecimal-string: A hexadecimal number (1 to FFFF) created based on the system time

The command returns `True` on successful execution, or `False` if an error occurs.

This command actually creates a zero-byte file with the specified file name.

Note that created files are not automatically deleted.

Note

Take care when specifying a folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Create a temporary file name and copy "MODEM.INF" to
' the created temporary file.
Dim bkupFileName, outDir1
outDir1 = _TEMP_
' Create a temporary file name with the prefix "BUP".
TempFile (bkupFileName, "BUP", outDir1)
Copy (outDir1+"MODEM.INF", bkupFileName)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.16 SetFileAttribute or SetFileAttr (set folder or file attributes)

Purpose

Sets folder or file attributes.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
SetFileAttribute (PathName, Attribute1 [, Attribute2, ...])
SetFileAttr (PathName, Attribute1 [, Attribute2, ...])
```

Arguments

PathName

Write the full path of the folder or file for which the attribute is to be set. Specify a character string or a variable that stores this value.

Attribute1 to *Attribute8*

Specify one or more of the following attributes. If you specify ATTR_NORMAL, you cannot specify any other attributes.

Value	Meaning
ATTR_READONLY	Read-only file
ATTR_HIDDEN	Hidden file
ATTR_ARCHIVE	Archive file
ATTR_SYSTEM	System file
ATTR_TEMPORARY	Temporary file You cannot specify this attribute if you specify a folder in <i>PathName</i> .
ATTR_NORMAL	Overrides all other attributes. This attribute can only be specified alone.

To add or remove only the attributes that you set in *Attribute1* to *Attribute8* for the folder or file specified in *PathName*, write either of the following values:

Value	Meaning
ATTR_ON (from version 05-10)	Add only the attributes set in <i>Attribute1</i> to <i>Attribute8</i> .
ATTR_OFF (from version 05-10)	Remove only the attributes set in <i>Attribute1</i> to <i>Attribute8</i> .

In JP1/Script 06-00 and later versions, you can set multiple attributes by specifying a one-dimensional array variable that stores each of the attributes.

Description

The `SetFileAttribute` or `SetFileAttr` command sets one or more attributes for the specified folder or file. The command returns `True` on successful execution, or `False` if an error occurs.

To acquire the attributes of a folder or file, use the `GetFileAttribute` command. To check one attribute of a folder or file, use the `IsFileAttribute` command.

Note

Take care when specifying the folder, or a folder or file in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Delete the archive attribute of a file. After accessing
' the file, check the archive attribute for file updates.
Dim file1
file1 = "C:\TEMP\logging.tmp"
If IsFileAttribute (file1 ,ATTR_ARCHIVE) = True Then
    SetFileAttribute (file1 ,ATTR_ARCHIVE ,ATTR_OFF)
End
...
(File access operations)
...
If IsFileAttribute (file1 ,ATTR_ARCHIVE) = True Then
    MessageBox (file1+ "has been updated.")
Else
    MessageBox (file1+ "was not updated.")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.17 GetFileAttribute or GetFileAttr (get folder or file attributes)

Purpose

Gets folder or file attributes.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
GetFileAttribute (PathName, AttrArrayBuff)
GetFileAttr (PathName, AttrArrayBuff)
```

Arguments

PathName

Write the full path of the folder or file for which you want to acquire attributes. Specify a character string or a variable that stores this value.

AttrArrayBuff

Specify the name of a dynamic one-dimensional array variable for storing the acquired attributes. The stored values may be any of the following:

Value	Meaning
ATTR_READONLY	Read-only file
ATTR_HIDDEN	Hidden file
ATTR_ARCHIVE	Archive file
ATTR_SUBDIR	Sub-directory
ATTR_SYSTEM	System file
ATTR_TEMPORARY	Temporary file
ATTR_COMPRESSED	Compressed file
ATTR_NORMAL	No particular attributes

The number of elements in the array variable are adjusted to the number of stored attributes.

Description

The `GetFileAttribute` or `GetFileAttr` command acquires the attributes set for the specified folder or file. The command returns `True` on successful execution, or `False` if an error occurs.

To set folder or file attributes, use the `SetFileAttribute` command. To check one attribute of a folder or file, use the `IsFileAttribute` command.

Note

Take care when specifying the folder, or a folder or file in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Delete all temporary files in the execution folder.
Dim file_attr()
For file = _SCF_+"*.*" Do
  ' Get file attributes.
  GetFileAttribute (file, file_attr)
  aryCnt = GetArrayCount (file_attr)
  ' Find temporary file attributes in the array variable
  ' that stores the file attributes.
  For i = 1 To aryCnt
    If file_attr(i) = ATTR_TEMPORARY Then
      ' Delete the file.
      DeleteFile (file, ExclDir)
    Exit For
  End If
Next
End For
```

JP1/Script version

Supported from JP1/Script 06-00.

8.5.18 SetFileTime (set file date and time)

Purpose

Sets the date and time of a file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
SetFileTime (PathName, [Date], [Time] [, Type])
```

Arguments

PathName

Write the full path of the file as a character string or as a variable that stores this value.

Date

Write the date to set in `yyyy/mm/dd` format as a character string or as a variable that stores this value. Numbers specified in the range 0 to 69 in `yyyy` are read as 2000 to 2069.

Time

Write the time to set in *hh:mm:ss* format as a character string or as a variable that stores this value.

Type

Specify the date type as either of the following values:

Value	Meaning
Create	Date created
Update	Date modified

This value is optional. If you omit this value, Update is assumed.

Description

The `SetFileTime` command sets the date and time of the specified file. The command returns `True` on successful execution, or `False` if an error occurs.

If you specify a file in the FAT system, you can only specify dates in the range 1980/1/1 to 2107/12/31.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
Dim file1
file1 = _SCF_ + "User.txt"
If IsFileAttr (file1, ATTR_ARCHIVE) Then
    SetFileAttr (file1, ATTR_ARCHIVE, ATTR_OFF)
    SetFileTime (file1, Date(), Time(), Update)
End If
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.19 GetFileTime (get file date and time)

Purpose

Gets the date and time of a file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
GetFileTime (PathName, [DateBuff], [TimeBuff] [, Type])
```

Arguments

PathName

Write the full path of the file as a character string or as a variable that stores this value.

DateBuff

Specify a variable for storing the date. The date is returned in *yyyy/mm/dd* format. Omit this argument if not required.

TimeBuff

Specify a variable for storing the time. The time is returned in *hh:mm:ss* format. Omit this argument if not required.

Type

Specify the date type to acquire as either of the following values:

Value	Meaning
Create	Date created
Update	Date modified

This value is optional. If you omit this value, Update is assumed.

Description

The `GetFileTime` command gets the date and time of the specified file. The command returns `True` on successful execution, or `False` if an error occurs.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
Dim file1, file2, dateBuff, timeBuff
file1 = _SCF_"User.txt"
file2 = _SCF_"UserBkup.txt"
If GetFileTime (file1, dateBuff, timeBuff, Create) = True Then
    SetFileTime (file2, dateBuff, timeBuff, Create)
End
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.20 GetFileSize (get file size)

Purpose

Gets the size of a file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
GetFileSize (PathName [, UnitofByte])
```

Arguments

PathName

Write the full path of the file as a character string or as a variable that stores this value.

UnitofByte

Specify the units in which to acquire the file size as one of the following values:

Value	Meaning
Byte	Bytes

Value	Meaning
KB	Kilobytes
MB	Megabytes

If you specify KB or MB in *UnitofByte*, the size is rounded up. For example, if you specify KB and the file size is less than 1 kilobyte, the command returns 1.

This value is optional. If you omit this value, `Byte` is assumed.

Even with a large file, if the conversion result exceeds 2,147,483,647, which is the maximum numeric value supported by JP1/Script, an error results.

Description

The `GetFileSize` command returns the size of the specified file. If an error occurs, zero is returned.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Check the file size, and copy the file if there is enough free space on
the disk.
Dim path1, path2, fileSz, freeSz
path1 = _SCF_+"Bkup.txt"
path2 = "A:"
fileSz = GetFileSize (path1)
freeSz = GetDiskFreeSpace (path2, KB)
If (freeSz * 1024) >= fileSz Then
    Copy (path1, path2)
End If
```

JP1/Script version

Supported from JP1/Script 06-00.

8.5.21 GetVersionInfo or GetVerInfo (get version information for a file)

Purpose

Gets version information for a file.

Syntax

```
GetVersionInfo (PathName, [Version1Buff], [Version2Buff],
[DescriptionBuff], [LegalCopyrightBuff], [CompanyNameBuff],
[OriginalFilenameBuff], [ProductVersionBuff], [ProductNameBuff] [,
InternalNameBuff])
GetVerInfo (PathName, [Version1Buff], [Version2Buff], [DescriptionBuff],
[LegalCopyrightBuff], [CompanyNameBuff], [OriginalFilenameBuff],
[ProductVersionBuff], [ProductNameBuff] [, InternalNameBuff])
```

Arguments

PathName

Write the full path of the file as a character string or as a variable that stores this value.

Version1Buff

Specify a variable for storing the file version in the form *9999.9999.9999.9999*. Omit this argument if not required.

The file version is returned as a character string in *9999.9999.9999.9999* form, where *9999* is a four-digit numerical string padded with leading zeros.

Version2Buff

Specify a variable for storing the file version. Omit this argument if not required.

The file version is returned as a string.

DescriptionBuff

Specify a variable for storing the description. Omit this argument if not required.

LegalCopyrightBuff

Specify a variable for storing the copyright. Omit this argument if not required.

CompanyNameBuff

Specify a variable for storing the company name. Omit this argument if not required.

OriginalFilenameBuff

Specify a variable for storing the original file name. Omit this argument if not required.

ProductVersionBuff

Specify a variable for storing the product version. Omit this argument if not required.

ProductNameBuff

Specify a variable for storing the product name. Omit this argument if not required.

InternalNameBuff

Specify a variable for storing the internal name. Omit this argument if not required.

Description

The `GetVersionInfo` or `GetVerInfo` command gets version information for the specified file and stores each item of information in the buffer variables. The command returns `True` on successful execution, or `False` if an error occurs.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Compare file versions.
Dim path1, ver1, ver2
path1 = "C:\Program Files\Hitachi\Script\Bin\SPTXE.EXE"
ver0520 = "0005.0020.0000.0000"
GetVersionInfo (path1, ver1, ver2)
If ver0520 < ver1 Then
    MsgBox ("File is later than" + ver0520 + _NL_ + _
           "file version:" + ver2)
End If
```

JP1/Script version

Supported from JP1/Script 06-00.

8.5.22 SplitFile (split a file)

Purpose

Partitions a file by a specified size.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
SplitFile (FilePath, SplitSize, [DirPath], [Option] [, SplitCnt])
```

Arguments

FilePath

Write the full path of the file as a character string or as a variable that stores this value.

SplitSize

Specify the split size as a number (bytes) or as a variable that stores this value.

Even with a large file, the maximum split size is 2,147,483,647 bytes.

DirPath

Specify a folder in which to store the split files, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the folder containing the file set in *FilePath* is assumed.

Option

Specify the following optional value:

Value	Meaning
Delete	Delete the file specified in <i>FilePath</i> after it is split.

SplitCnt

Specify a variable for storing the number of split files. Omit this argument if not required.

Description

The `SplitFile` command partitions a files by the specified size and stores the split files in the specified folder. The command returns `True` on successful execution, or `False` if an error occurs.

The file names of the split files have the file name specified in *FilePath*, plus the extension `.xxx` (where `xxx` is a number from 001 to 999, then restarting at 1000).

Notes

- Take care when specifying a file or folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions for split files. In the following registry key, set the access permissions beforehand.

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option
```

Value name

```
SecurityAttributesSucceed
```

Value datatype

```
REG_DWORD
```

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

```
' Partition the file "C:\FDCOPY.TXT" to fit on 1.44-MB
' floppy disks and store the split files on floppy disks.
' (Ref) 1.44MB FD = 1423KB = 1423 * 1024B
'       1.25MB FD = 1221KB = 1221 * 1024B
'       720KB  FD = 713KB  = 713 * 1024B
'       640KB  FD = 640KB  = 640 * 1024B
Dim file1, size1, splCnt, filePath
file1 = "FDCOPY.TXT"
size1 = 1423 * 1024
SplitFile ("C:\"+file1, size1, _TEMP_, , splCnt)

Dim cnt1

' Create floppy disks for the number of split files.
For cnt1 = 1 To splCnt
    filePath = _TEMP_+file1
    Select Case Len (cnt1)
        Case 1
            filePath = filePath+".00"+cnt1
        Case 2
            filePath = filePath+".0"+cnt1
        Case Else
            filePath = filePath+"."+cnt1
    End Select

    ' Copy the temporary files to floppy disks.
    Copy (filePath, "A:\", Overwrite)
    If splCnt - cnt1 <= 0 Then
        MsgBox ("Copying to floppy disk has ended.", OK)
        Exit For
    End
    MsgBox ("Insert floppy disk number" + cnt1+1", _
            OKCancel)
    If _MSG_RTN_ = Cancel Then
        Exit For
    End
Next
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.23 CatFiles (join split files)

Purpose

Joins split files into one file.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
CatFiles (PathName, [Option], FilePath1, [FilePath2] [, FilePath3 ,) ])
```

Arguments

PathName

Specify the folder or file in which to save the joined file, using a character string or a variable that stores this value. If you specify an existing file, its contents will be replaced.

Option

Specify the following optional value:

Value	Meaning
Delete	Delete the files specified in <i>FilePathn</i> arguments after they are joined.

FilePathn

Write the full path of each file to be joined, using a character string or a variable that stores this value. You can use a wildcard in the file names.

The files are joined in the order specified, to a maximum of 10 files.

In JP1/Script 06-00 and later versions, you can specify a one-dimensional array variable that stores strings representing the full path of each file. This method allows more than 10 files to be specified.

If you specify a wildcard in a file name, the matching file names are sorted and joined in ascending order.

Description

`CatFiles` joins the specified files and stores them in the specified folder. The command returns `True` on successful execution, or `False` if an error occurs.

Notes

- Take care when specifying a file or folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating a file to store the joined files. In the following registry key, set the access permissions beforehand:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option
```

Value name

```
SecurityAttributesSucceed
```

Value datatype

```
REG_DWORD
```

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Important note

The following types of file names are used in a Windows file system:

- Long file names with character strings specified by a user
- Short file names automatically generated by Windows (in 8.3 format)

Therefore, if a wildcard is specified in a file name in JP1/Script, the short file names automatically generated by the OS are also included. In addition, if a wildcard is not specified for a file name, the short file name of another file that exists in the same path as the specified file might be the same, resulting in operations being performed on files you did not intend.

Example

```
' Join the files "A:\FDCOPY.TXT.*" saved to floppy disk,  
' and create a joined file "C:\SCRIPT\FDCOPY.TXT".  
Dim dir1, file1  
dir1 = "C:\SCRIPT\  
file1 = "FDCOPY.TXT"  
CatFiles (dir1+file1, , "A:\"+file+"*")
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.24 SetStandardFile or SetStdFile (set the standard input, output, or error file)

Purpose

Opens and sets the standard input, standard output, or standard error file for processes called by the `Exec` command.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
SetStandardFile (FilePath, Type [, Mode])  
SetStdFile (FilePath, Type [, Mode])
```

Arguments

FilePath

Write the full path of the file to be set, using a character string or a variable that stores this value.

Type

Specify the file type as one of the following values:

Value	Meaning
StdInput	Standard input
StdOutput	Standard output
StdError	Standard error

Mode

If you specified `StdOutput` or `StdError` in *Type*, specify the creation mode as one of the following values:

Value	Meaning
Create	Create a new file.
Append	Add data to the existing file.

This value is optional. If you omit this value, Append is assumed.

This value is invalid if you specified StdInput in *Type*.

Description

The `SetStandardFile` or `SetStdFile` command opens and sets the standard input, standard output, or standard error file for processes called subsequently by the `Exec` command. The command returns `True` on successful execution, or `False` if an error occurs.

The set files remain effective and open until execution of the `ResetStandardFile` command.

The files set by this command cannot be accessed by the `NetExec` command.

Notes

- Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating a file you are setting. In the following registry key, set the access permissions beforehand:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option

Value name

SecurityAttributesSucceed

Value datatype

REG_DWORD

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note

If any of the standard files (standard input, standard output, or standard error) is specified, processes executed by the `Exec` command may not execute normally. If this occurs, prepare empty files, and specify all three standard files (standard input, standard output, and standard error).

Example

```
' When Backup.BAT is executed, write standard output in
' BatOut.TXT, and write standard error output in BatErr.TXT.
SetStandardFile (_SCF_"BatOut.TXT", StdOutput, Create)
SetStandardFile (_SCF_"BatErr.TXT", StdError)
If Exec (_SCF_"Backup.BAT", True) Then
    MessageBox ("Exec command succeeded.")
Else
    MessageBox ("Exec command failed.")
```



```
End
ResetStandardFile ( StdOutput )
ResetStandardFile ( StdError )
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.25 ResetStandardFile or ResetStdFile (reset the standard input, output, or error file)

Purpose

Disconnects the standard input, standard output, or standard error file for processes called by the `Exec` command. In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
ResetStandardFile ([Type])
ResetStdFile ([Type])
```

Argument

Type

Specify the type of file to reset, using one of the following values:

Value	Meaning
StdInput	Standard input
StdOutput	Standard output
StdError	Standard error

This value is optional. If you omit this value, all the standard files are reset.

Description

The `ResetStandardFile` or `ResetStdFile` command closes and disconnects the standard input, standard output, or standard error file set by the `SetStandardFile` command. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
SetStandardFile (_SCF_+"BatOut1.TXT", StdOutput)
SetStandardFile (_SCF_+"BatErr.TXT", StdError)
Exec (_SCF_+"Backup.BAT", True)

ResetStandardFile (StdOutput)
SetStandardFile (_SCF_+"BatOut2.TXT", StdOutput)
Exec (_SCF_+"Build.BAT", True)
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.26 SplitPath (analyze a full path)

Purpose

Analyzes a full path.

Syntax

```
SplitPath (FullPath, [DrvNameBuff], [DirNameBuff ], [LblNameBuff] [, ExtNameBuff])
```

Arguments

FullPath

Write the full path of the file as a character string or as a variable that stores this value.

DrvNameBuff

Specify a variable for storing the drive name. Omit this argument if not required.

DirNameBuff

Specify a variable for storing the folder name. Omit this argument if not required.

LblNameBuff

Specify a variable for storing the file name. Omit this argument if not required.

ExtNameBuff

Specify a variable for storing the file extension. Omit this argument if not required.

Description

The `SplitPath` command analyzes the specified full path into the drive name, folder name, file name, and file extension, and stores each part in the buffer variables. The command returns `True` on successful execution, or `False` if an error occurs.

A colon (:) is appended to the drive name. The folder name is enclosed with backslashes (\). The extension is preceded by a period (.).

Example

```
Dim drv1, dir1, lbl1, ext1
SplitPath ("C:\WINDOWS\WINFILE.EXE", drv1, dir1, lbl1, ext1)

drv1 C:
dir1 \WINDOWS\
lbl1 WINFILE
ext1 .EXE
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.27 MakePath (create a full path)

Purpose

Creates a full path.

Syntax

```
MakePath (FullPathBuff, DrvName, DirName, LblName, ExtName)
```

Arguments

FullPathBuff

Specify a variable for storing the created full path.

DrvName

Specify the drive name as a character string or as a variable that stores this value.

DirName

Specify the folder name as a character string or as a variable that stores this value.

LblName

Specify the file name as a character string or as a variable that stores this value.

ExtName

Specify the file extension as a character string or as a variable that stores this value.

Description

The `MakePath` command creates a full path by joining the drive name, folder name, file name, and file extension, and stores the full path in the buffer variable. The command returns `True` on successful execution, or `False` if an error occurs.

The command automatically appends a colon (:) to the drive name, encloses the folder name with backslashes (\), and prefixes a period (.) to the extension even if you omit any of these symbols.

Example

```
' Analyze the file name stored in OriginalFile, and
' create a full path with new extension ".BAK" in BackupFile.
Dim OriginalFile ,BackupFile ,Drv ,Dir ,Lbl ,Ext
OriginalFile = _SCF_ + "TEST.SPT"
SplitPath ( OriginalFile ,Drv ,Dir ,Lbl ,Ext )
MakePath ( BackupFile ,Drv ,Dir ,Lbl ,".BAK" )
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.28 SetPath (set the path to the executable folder)

Purpose

Sets the path to the executable folder.

Syntax

```
SetPath ([DirPath])
```

Argument

DirPath

Specify the folder path to be set, using a character string or a variable that stores this value.

This value is optional. If you omit this value, the current folder (work folder) is assumed.

Description

The `SetPath` command sets the specified folder path as the current folder (work folder) path. The `_BIN_` reserved variable is also converted to the specified folder path.

The command returns `True` on successful execution, or `False` if an error occurs. If the specified folder path does not exist, the command always returns `False` without changing the current folder (work folder) path.

Example

```
' Set the current folder for executable file "ABC.EXE" and
' execute the file.
' The current folder is OutDir folder specified by argument
' if OutDir folder contains a file.
' If OutDir folder is empty, the current folder is the folder
' of the script file being executed.
OutDir = %1
If IsEmptyDir ( OutDir ) = False Then
    SetPath ( OutDir )
    Exec ( "ABC.EXE" , True )
    SetPath
Else
    Exec ( _SCF_+"ABC.EXE" , True )
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.29 GetPath (get the path to the executable folder)

Purpose

Gets the path to the executable folder.

Syntax

```
GetPath
```

This command has no arguments.

Description

The `GetPath` command gets the current path to the executable folder and returns the path as the execution result. The command does not append a backslash (\) to the folder name unless the root folder is the executable folder. If you want to have a backslash at the end of the folder name, specify 1 in the following registry:

Registry key

```
HKEY_LOCAL_MACHINE\Software\Hitachi\JP1/Script\SPTX
```

Value name

```
GetPath_Opt
```

Value datatype

```
REG_DWORD
```

Value

- 0: Do not add a backslash (\) to the folder name (initial value).
- 1: Add a backslash (\) to the folder name.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

```
' If "Script.ini" exists,  
' copy it to the "BKUP" folder in the current folder.  
outDir = GetPath + "\BKUP\  
If IsExistFile( "Script.ini" ) Then  
  Copy ( "Script.ini" , outDir )  
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.5.30 SetVolumeLabel or SetVolLabel (set a disk volume label)

Purpose

Sets the volume label of a disk.

Syntax

```
SetVolumeLabel (DiskName [, LabelName])  
SetVolLabel (DiskName [, LabelName])
```

Arguments

DiskName

Specify the disk volume as a character string or as a variable that stores this value. You can omit the colon (:).

LabelName

Specify the volume label to set, using a character string or a variable that stores this value.

This value is optional. If you omit this value, a zero-length string ("") is assumed.

Description

The `SetVolumeLabel` or `SetVolLabel` command sets the volume label of a disk. The command returns `True` on successful execution, or `False` if an error occurs.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
Dim drvName, volName  
drvName = "E:"  
volName = GetVolumeLabel (drvName)  
If volName = Empty Then  
  SetVolumeLabel (drvName, "Script shared drive")  
End
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.31 GetVolumeLabel or GetVolLabel (get a disk volume label)

Purpose

Gets the volume label of a disk.

Syntax

```
GetVolumeLabel (DiskName [, FileSystemBuff])  
GetVolLabel (DiskName [, FileSystemBuff])
```

Arguments

DiskName

Specify the disk volume as a character string or as a variable that stores this value. You can omit the colon (:).

FileSystemBuff

Specify a variable for storing the acquired file system name (FAT, HPFS, NTFS, or other name). Omit this argument if not required.

Description

The `GetVolumeLabel` or `GetVolLabel` command gets the volume label of the specified disk, and returns the label as the execution result.

If you specify a non-existent disk, the command returns a zero-length string ("").

Example

```
Dim volName, fileSystem  
volName = GetVolumeLabel ("E:", fileSystem)
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.32 GetDiskFreeSpace (get the disk free space)

Purpose

Gets the amount of free space on a disk.

Syntax

```
GetDiskFreeSpace (DiskName [, UnitofByte])
```

Arguments

DiskName

Specify the disk volume as a character string or as a variable that stores this value. You can omit the colon (:).

UnitofByte

Specify the units in which to acquire the free space as one of the following values:

Value	Meaning
KB	Kilobytes
MB	Megabytes

The returned free space size is rounded down. For example, if you specify MB and the file size is less than 1 megabyte, the command returns zero.

This value is optional. If you omit this value, KB is assumed.

Description

The `GetDiskFreeSpace` command calculates the free space on the specified disk and returns this value as the execution result.

Zero is returned if an error occurs.

Example

```
Dim MBSize, Bsize
MBSize = GetDiskFreeSpace ("A:", MB)
Bsize = MBSize * 1024 * 1024
MessageBox ("Floppy disk capacity is" +Bsize +"bytes.")
```

JP1/Script version

Supported from JP1/Script 05-10.

8.5.33 Copy (copy files)

Purpose

Copies files.

In JP1/Script 07-50 or later, the command supports large files.

Syntax

```
Copy (OldFileName, NewPathName, [Option1], [Option2], [Option3],
[Option4], [Option5], [ExceptFileName], [Option6] [, Option7])
```

Arguments

OldFileName

Set the name of the file to be copied (or moved). Write a character string or a variable that stores this value.

If the specified file name ends with an underscore (`_`), the file is assumed to have been compressed with the `COMPRESS` command, and will be expanded when copied.

If you did not specify a new file name in the *NewPathName* argument, the file name remains unchanged. The final character remains as an underscore (`_`). You can specify a wildcard in the file name.

NewPathName

Set the name of the folder or file at the copy (or move) destination. Write a character string or a variable that stores this value.

Option1 to Option5

Specify one or more of the following copy options:

Value	Meaning
<code>VersionUp</code>	Overwrite new files only. Compare the version information. If version information cannot be obtained, compare the dates of the files.
<code>Overwrite</code>	Overwrite all files.
<code>NoOverwrite</code> (from 05-10)	Do not copy files that already exist in the destination folder.
<code>OverwriteOnly</code> (from 05-10)	Copy only files that already exist in the destination folder.

Value	Meaning
Pile	Change extension values of existing files to unused values from .000 to .999 and copy files.
Backup (from 05-10)	Copy only files that have the archive attribute. After copying, clear the archive attribute of the source files.
Move (from 05-10)	Move the files. Do not copy them. You can specify <code>Move</code> in combination with <code>VersionUp</code> , <code>Overwrite</code> , or <code>NoOverwrite</code> . If a file already exists in the destination folder, the source file is moved according to the options you specify.
SubDirToo (from 05-10)	Copy the files and sub-folders, preserving the tree structure. This option does not copy any sub-folder that contains no files. You can specify <code>SubDirToo</code> with other copy options.
Trace (from 05-10)	Output the copied file names to the execution trace file. You can specify <code>Trace</code> with other copy options.
ErrSkip (from 05-10)	Continue copying regardless of any lock errors or access errors at a file being copied. You can specify <code>ErrSkip</code> with other copy options.

This value is optional. If you omit this value, `Overwrite` is assumed.

ExceptFileName (from version 05-10)

If the file name you set in *OldFileName* includes a file that you do not want to copy or move, specify that file name as a character string or as a variable that stores that value.

To set multiple file names here, write the file names as a semicolon- (;) delimited string or as a variable that stores this value.

In JP1/Script 06-00 and later versions, you can specify a one-dimensional array variable that stores strings representing multiple file names.

You can also specify a wildcard in *ExceptFileName*.

Option6 and *Option7* (from version 06-51)

Specify either of the following optional values:

Value	Meaning
Security	Copy any security information attached to the file. You can specify <code>Security</code> with other copy options.
ErrSkip2 (from 06-71)	Continue copying regardless of any lock errors or access errors on a destination file. The number of files with skipped errors is stored in the <code>_COPY_SKIP2_CNT_</code> reserved variable. You can specify <code>ErrSkip2</code> with other copy options.

If you specify `Security`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Description

The `Copy` command copies files according to the specified copy options. The command returns `True` on successful execution, or `False` if an error occurs.

The `_COPY_RTN_` reserved variable stores one of the following as the copy result:

Value	Meaning
Overwrite	The files were copied without a file version check.
VersionUp	The files were copied after checking the file version.
Pile	The files were copied, and existing versions were preserved with a different extension.
Backup (from 05-10)	Only files with the archive attribute were copied, and the archive attribute was subsequently cleared.
Move (from 05-10)	The files were moved.

Value	Meaning
Skip	The files were not copied.

The number of files copied is stored in the `_COPY_CNT_` reserved variable. The number of files that were not copied is stored in the `_COPY_SKIP_CNT_` reserved variable.

Notes

- Take care when specifying a file or folder in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating a copy destination file. In the following registry key, set the access permissions beforehand:
However, you cannot configure access permissions if `Security` is specified for the copy options `Option6` and `Option7`.

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script/SPTX/Option

Value name

SecurityAttributesSucceed

Value datatype

REG_DWORD

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Important note

The following types of file names are used in a Windows file system:

- Long file names with character strings specified by a user
- Short file names automatically generated by Windows (in 8.3 format)

Therefore, if a wildcard is specified in a file name in JP1/Script, the short file names automatically generated by the OS are also included. In addition, if a wildcard is not specified for a file name, the short file name of another file that exists in the same path as the specified file might be the same, resulting in operations being performed on files you did not intend.

Example

```
Dim inDir, outDir
inDir = _SCF_+"Inst\"
outDir = _SCF_+"Inst_Backup\"
Copy (inDir+"CTL3D32.DLL", outDir+"CTL3D32.DLL", VersionUp)
Copy (inDir+"SCRIPT.EX_", outDir+" SCRIPT.EXE", Overwrite)
Copy (inDir+"SCR*.*", outDir)
```

```
Copy (inDir, outDir, Backup, ErrSkip, , , _  
, inDir+"*.EXE;" + inDir+"*.DLL")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.6 Commands for message output

8.6.1 InputBox (display a message and text boxes)

Purpose

Displays a message and text boxes in a dialog box, and returns the text box contents when the user enters text or clicks a button.

Syntax

```
InputBox ([Text], [Title], [xPos], [yPos], InputBuff1, [Caption1],  
[InputBuff2], [Caption2], ...)
```

Arguments

Text

Specify the message string to be displayed in the dialog box. Write a character string or a variable that stores this value. If #Option = NOCHANGE is missing from the head of the script file, any \r, \n, \t, or \\ strings included in the message text are processed as control codes.

For details about control codes, see [6.1.11 Script coding conventions](#).

Title

Specify the title to be displayed in the title bar as a character string or as a variable that stores this value.

This value is optional. If you omit this value, **JP1/Script InputBox** appears in the title bar.

xPos

Specify the horizontal distance from the left side of the screen to the left edge of the dialog box. Specify the distance in pixels, where the left side of the screen is 0, or write a variable that stores this value. If you omit this argument, the dialog box is centered horizontally on the screen. If you specify a negative value, zero is assumed.

yPos

Specify the vertical distance from the top of the screen to the top of the dialog box. Specify the distance in pixels, where the top of the screen is 0, or write a variable that stores this value. If you omit this argument, the dialog box is centered vertically on the screen. If you specify a negative value, zero is assumed.

InputBuff1 to InputBuff4

Specify variables for storing the strings that the user types in the text boxes. You can specify up to four text boxes. If you preset a value in a buffer variable, that value will appear in the text box as its initial value. If the user does not enter anything in an text box, a zero-length string ("") is stored in the associated buffer variable.

Caption1 to Caption4

Specify a caption for each of the text boxes you set in *InputBuff1* to *InputBuff4*. Write each caption as a string of up to 40 characters, or as a variable that stores this value.

This value is optional. If you omit this value, the text box will have no caption.

Description

The `InputBox` command displays a message and up to four text boxes according to the specified parameters. The command returns `True` on successful execution, or `False` if an error occurs.

An **OK** button and **Cancel** button are added to the message box. When the user presses the **OK** button or the **Enter** key, the command returns the contents of the text boxes. If the user presses the **Cancel** button, the command returns zero-length strings ("").

The value of the button that the user chooses is stored in the `_MSG_RTN_` reserved variable as either of the following:

Value	Button chosen by the user
OK	OK
CANCEL	Cancel

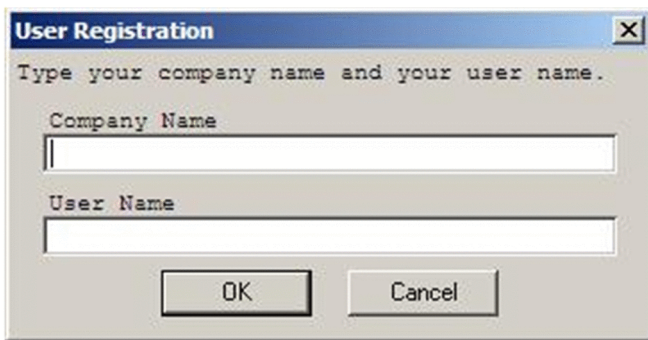
Note

You cannot use this command in a script started as a service. An execution error occurs if this command is used.

Example

```
Dim compName
Dim userName
InputBox ("Type your company name and your user name.", _
    "User Registration", 100, 100, compName, _
    "Company Name", userName, "User Name")
```

Display example



JP1/Script version

Supported from JP1/Script 01-00.

8.6.2 Message (output text to a file or window)

Purpose

Outputs specified text to a file or window. Also erases the displayed window and message text.

Window display is supported from JP1/Script 01-01.

Syntax

```
Message (Target, [OutputName], [Text], [LineLength], [MaxLines], [xPos]
[, yPos])
```

Arguments

Target

Specify the text output destination as one of the following values:

Value	Meaning
Target_File	Output to a file.
Target_Dispon (from 01-01)	Output to a window.
Target_Dispclear (from 01-01)	Erase the message text displayed in the window.

Value	Meaning
Target_DispOff (from 01-01)	Erase the open window.
Target_SPAMFile (from 05-20)	Output to the executing script's analysis trace file.
Target_SPXFile (from 05-20)	Output to the executing script's execution trace file.

OutputName

- If you specified `Target_File` in *Target*:
Specify the full path of the output file as a character string or as a variable that stores this value. If you omit the file extension, `.TXT` is automatically appended to the file name. However, in JP1/Script 06-00 and later versions, if you specify a file name ending with the extension period (`.`), the file is regarded as having no extension.
- If you specified `Target_DispOn`, `Target_DispClear`, or `Target_DispOff` in *Target*:
Specify the window title as a character string or as a variable that stores this value.
- If you specified `Target_SPAMFile` or `Target_SPXFile` in *Target*:
You can omit the *OutputName* value because the system automatically sets the executing script's analysis trace file or execution trace file for trace output. If you specify *OutputName*, the system takes the specified string as a window title and outputs the message text to both the trace file and the window.

Text

- If you specified `Target_File`, `Target_DispOn`, `Target_SPAMFile`, or `Target_SPXFile` in *Target*:
Specify the message text to output. Write a character string or specify a variable that stores this value. If `#Option = NOCHANGE` is missing from the head of the script file, any `\r`, `\n`, `\t`, or `\\` strings included in the message text are processed as control codes.
For details about control codes, see [6.1.11 Script coding conventions](#).
- If you specified `Target_DispClear` or `Target_DispOff` in *Target*:
The *Text* argument is invalid.

LineLength

- If you specified `Target_File` in *Target*:
Specify the line length of the output message text, excluding any `\r` and `\n` codes included in the message. Write the number of bytes or specify a variable that stores this value.
You only need to specify a *LineLength* value the first time you execute the `Message` command for the file specified in *OutputName*. If you specify a different line length at any subsequent execution of the `Message` command for the same output file, the file will be recreated with the new line length.
You can specify a value in the range from 128 to 1,024.
The *LineLength* value is optional. If you omit this value, the value set in **Max columns** in **User trace information** in the execution environment file is assumed. If you omit this value and user trace information does not exist, the system assumes 150.
- If you specified `Target_DispOn`, `Target_DispClear`, or `Target_DispOff` in *Target*:
The *LineLength* value is invalid.
- If you specified `Target_SPAMFile` or `Target_SPXFile` in *Target*:
The *LineLength* value is invalid. The system assumes the value set in **Max columns** in **User trace information** in the execution environment file.

MaxLines

- If you specified `Target_File` in *Target*:

Specify the maximum lines in the output message text as a number or as a variable that stores this value.

As with *LineLength*, you only need to specify a *MaxLines* value the first time you execute the `Message` command for the file specified in *OutputName*. If you specify a different value in *MaxLines* at any subsequent execution of the `Message` command for the same output file, the file will be recreated with the new maximum number of lines.

You can specify a value in the range from 100 to 9,999.

The *MaxLines* value is optional. If you omit this value, the system assumes the value set in **Max lines** in **User trace information** in the execution environment file. If you omit this value and user trace information does not exist, the system assumes 1,024.

- If you specified `Target_Dispon`, `Target_Dispclear`, or `Target_Dispoft` in *Target*:

The *MaxLines* value is invalid.

- If you specified `Target_SPaFile` or `Target_SPXFile` in *Target*:

The *MaxLines* value is invalid. The system assumes the value set in **Max lines** in **User trace information** in the execution environment file.

xPos

- If you specified `Target_Dispon` or `Target_Dispclear` in *Target*, or specified `Target_SPaFile#1` or `Target_SPXFile#1` in *Target* in order to output trace messages to a window as well as to a trace file:

Specify the horizontal distance from the left side of the screen to the left edge of the window. Specify the distance in pixels, where the left side of the screen is 0, or write a variable that stores this value.

If you omit this argument, the window is centered horizontally on the screen. If you specify a negative value, zero is assumed.

- If you specified `Target_File` or `Target_Dispoft` in *Target*, or specified `Target_SPaFile#2` or `Target_SPXFile#2` in *Target* in order to not output trace messages to a window:

The *xPos* value is invalid.

yPos

- If you specified `Target_Dispon` or `Target_Dispclear` in *Target*, or specified `Target_SPaFile#1` or `Target_SPXFile#1` in *Target* in order to output trace messages to a window as well as to a trace file:

Specify the vertical distance from the top of the screen to the top of the window. Specify the distance in pixels, where the top of the screen is 0, or write a variable that stores this value.

If you omit this argument, the window is centered vertically on the screen. If you specify a negative value, zero is assumed.

- If you specified `Target_File` or `Target_Dispoft` in *Target*, or specified `Target_SPaFile#2` or `Target_SPXFile#2` in *Target* in order to not output trace messages to a window:

The *yPos* value is invalid.

^{#1} When a value is specified in *OutputName*

^{#2} When no value is specified in *OutputName*

Description

The `Message` command outputs a specified message text to a specified file or window. The command may also erase the displayed window and text message.

The command returns `True` on successful execution, or `False` if an error occurs.

Notes

- You cannot use this command in a script started as a service. An execution error occurs if this command is used.
- Take care when specifying an initialization file in the folder set in the environment variable ProgramFiles (normally the Program Files folder on the system drive) or WinDir (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).
- You can configure access permissions when creating a file specified for Target_File. In the following registry key, set the access permissions beforehand:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option

Value name

SecurityAttributesSucceed

Value datatype

REG_DWORD

Value

0: Access permissions are not set.

1: Access permissions are set to "Inherit access permissions from parent folders".

2: Access permissions are set to "Everyone: Full control".

If no value is set, or a value other than those above is set, the setting defaults to 0.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Important note

- If a script that specifies multi-activation in the execution environment contains a Message command that specifies Target_SPXFile or Target_SPAFile, and that script file is executed several times concurrently, the messages are output to the same analysis trace file or execution trace file. Note that because of this, exclusion control is applied to that file and execution performance may drop. The same situation occurs when the file name that is output by Target_File is fixed.
- The files output by the Message command are not processed by the log file trapping function of JP1/Cm2/Extensible SNMP Agent or JP1/Base.
- If you created files by using the Message command with Target_File specified, do not output these files in text format by using commands for manipulating files and folders or by using other applications. The output result of the subsequent Message command cannot be guaranteed.
- If you create a large number of user trace files with unique names by using the Message command with Target_File specified, the increase in trace management files may have adverse effects on the performance of script execution. Script execution may terminate abnormally due to insufficient memory, or command execution may result in an error as shown in the following examples:
 - Script execution terminates abnormally with termination code 20.
 - The Copy command displays the message Insufficient memory.

In this case, use the TextOpen, TextWrite, or TextClose command instead of the Message command to create a trace file.

Example

```
' Write the script execution log to "Logging.txt".  
Message (Target_File, _BIN_"Logging.txt", "Execution_
```

```

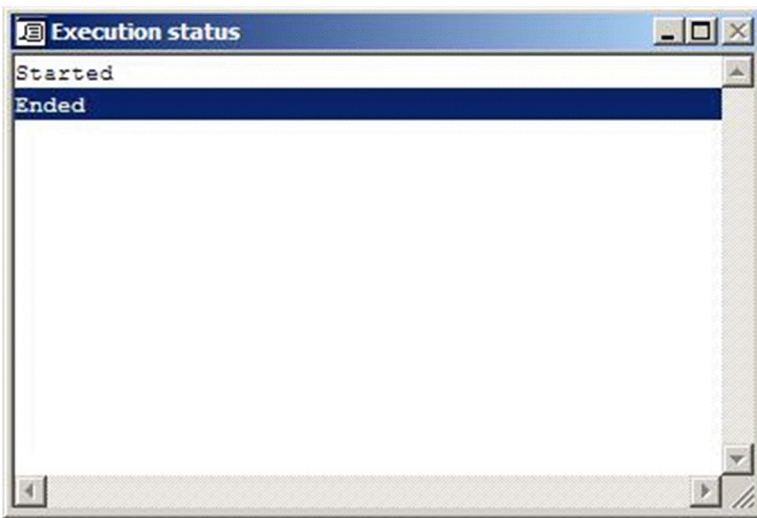
        has started.", 30, 100)
    ...
    Message (Target_File, _BIN_"Logging.txt", "Execution_
        has ended.")

    ' Display the script execution history on the screen.
    Message (Target_Dispon, "Execution status", "Started", _
        , , 100, 100)

    ...
    Message (Target_Dispon, "Execution status", "Ended", , _
        , 100, 100)
    Sleep (3000)
    Message (Target_Dispon, "Execution status")

```

Display example



JP1/Script version

Supported from JP1/Script 01-00.

8.6.3 MessageBox (display a message in a dialog box)

Purpose

Displays a specified message in a dialog box. Buttons or icons may be added. When the box has buttons, the command returns a value indicating which button the user clicked.

Syntax

```

MessageBox (Text, [Buttons], [DefaultBtn], [IconStyle], [ModalStyle] [,
    Title])

```

Arguments

Text

Specify the message string to be displayed in the dialog box. Write a character string or a variable that stores this value. If #Option = NOCHANGE is missing from the head of the script file, any \r, \n, \t, or \\ strings included in the message text are processed as control codes.

For details about control codes, see [6.1.11 Script coding conventions](#).

Buttons

Specify the types and number of buttons to add to the dialog box as one of the following values:

Value	Meaning
OK	Display an OK button only.
OKCancel	Display OK and Cancel buttons.
YesNo	Display Yes and No buttons.
YesNoCancel	Display Yes , No , and Cancel buttons.
RetryCancel	Display Retry and Cancel buttons.
AbortRetryIgnore or Abort	Display Abort , Retry , and Ignore buttons.

This value is optional. If you omit this value, the dialog box will have an **OK** button only.

DefaultBtn

Specify which of the buttons you set in the *Buttons* argument is to be the default button. Enter one of the following values:

Value	Meaning
1	Make the first button the default.
2	Make the second button the default.
3	Make the third button the default.

This value is optional. If you omit this value, 1 is assumed.

IconStyle

Specify the style of the icon to add to the dialog box as one of the following values:

Value	Meaning
Exclamation or Ex	Exclamation mark (!)
Information or Info	Letter i inside a circle
Question	Question mark (?)
Stop	STOP

This value is optional. If you omit this value, the dialog box will have no icon.

ModalStyle

Specify whether the dialog box is modal. Use either of the following values:

Value	Meaning
ApplicationModal or AppliModal	Set the dialog box to Application Modal. The user can move and work in a window in another application.
SystemModal or SysModal	Set the dialog box to System Modal. Use this value for notifying the user of a critical error that requires immediate attention.

This value is optional. If you omit this value, `ApplicationModal` is assumed.

Title

Specify the title to be displayed in the title bar as a character string or as a variable that stores this value.

This value is optional. If you omit this value, **JP1/Script MessageBox** appears in the title bar.

Description

The `MessageBox` command displays a message box according to the specified parameters. The command returns `True` on successful execution, or `False` if an error occurs.

The value of the button that the user chooses is stored in the `_MSG_RTN_` reserved variable as either of the following:

Value	Button chosen by the user
OK	OK
CANCEL	Cancel
ABORT	Abort
RETRY	Retry
IGNORE	Ignore
YES	Yes
NO	No

In a dialog box with a **Cancel** button, pressing the **Esc** key has the same effect as clicking **Cancel**.

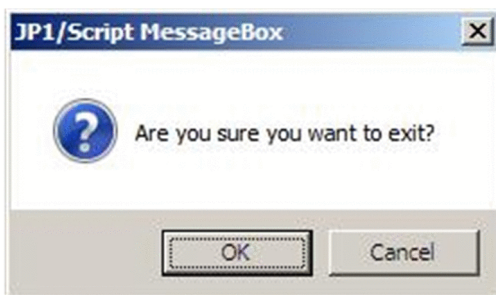
Note

You cannot use this command in a script started as a service. An execution error occurs if this command is used.

Example

```
MessageBox ("Are you sure you want to exit?", OKCancel, , _
           Question)
If _MSG_RTN_ <> CANCEL Then
    Exit
End
```

Display example



JP1/Script version

Supported from JP1/Script 01-00.

8.6.4 MessageEventLog (output a message to the application log in Event Viewer)

Purpose

Outputs a message to the application log in Event Viewer.

Syntax

```
MessageEventLog (Text, [LogType] [, Option])
```

Arguments

Text

Specify the message to be output to Event Viewer. Write a character string or a variable that stores this value. If #Option = NOCHANGE is missing from the head of the script file, any \r, \n, \t, or \\ strings included in the message text are processed as control codes.

For details about control codes, see [6.1.11 Script coding conventions](#).

LogType

Specify the type of events for message output as one of the following values:

Value	Meaning
Error or Err	Error
Warning	Warning
Information or Info	Information
AuditSuccess	Audit successful
AuditFailure	Audit failed

This value is optional. If you omit this value, Information is assumed.

Option (from version 06-51)

Specify the following optional value:

Value	Meaning
TextOnly	Do not prefix the message with the user's name and executed script file name. Output only the message specified in Text.

Description

The MessageEventLog command outputs a message to the application log at Event Viewer according to the specified parameters. The command returns True on successful execution, or False if an error occurs.

Messages output using this command have 3 as their event ID. Unless you specify TextOnly in the Option argument, the output messages are prefixed with the user's name and executed script file name.

The MessageEventLog command is executed even if the parameter /NOEVLOG or /NOEVLOG (3) is set in the command line defined in the execution environment file or in the command line set in the registry.

Example

```
If Exec ("ABC.EXE", True) = False Then
    MessageEventLog ("ABC.EXE call failed.", Error)
End
```

JP1/Script version

Supported from JP1/Script 01-01.

8.6.5 IMEventMessage (issue events to JP1/IM or JP1/Base)

Purpose

Issues events to JP1/IM or JP1/Base.

Syntax

```
IMEventMessage (Text, [Severity], [UserName], [RootObjName],  
[Occurrence], [StartDate], [StartTime], [EndDate], [EndTime],  
[ResultCode])
```

Arguments

Text

Specify the message to be displayed as an event at the JP1/IM Event Console. Write a character string or a variable that stores this value.

Severity

Specify the severity of the event as one of the following values:

Value	Meaning
Emergency or Emergence	Emergency: Panic state Note: If you specify Emergence instead of Emergency, the automatic actions of JP1/IM do not operate because they are not considered JP1 events.
Alertness	Warning: Prompt action required
Critical	Critical: Major error
Error or Err	Error in the application
Warning	Warning message
Notice	Notice: Not an error, but special processing required
Information or Info	Information message
Debug	Debug: Information used for debugging a program

This value is optional. If you omit this value, `Information` is assumed in the **Severity** field at the Event Console.

UserName

Specify the user name of the event as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a blank is set in the **User Name** field at the Event Console.

RootObjName

Specify the registered object name of the event as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a blank is set in the **Object Name** field at the Event Console.

Occurrence

Specify the event type as one of the following values:

Value	Meaning
Start	Execution started.
End	Execution ended.
Create	Definition created.
Destroy	Definition deleted.

Value	Meaning
Modify	Definition modified.
Submit	Submitted for execution
UnSubmit	Removed from executable status.
Pause	Execution temporarily suspended.
Release	Execution resumed.
Restart	Execution restarted.

This value is optional. If you omit this value, a blank is set in the **Event Type** field at the Event Console.

StartDate

Specify the event start date in *yyyy/mm/dd* format.

This value is optional. If you omit this value, a blank is set in the **Start Date** field at the Event Console.

StartTime

Specify the event start time in *hh:mm:ss* format.

This value is optional. If you omit this value, a blank is set in the **Start Time** field at the Event Console.

EndDate

Specify the event end date in *yyyy/mm/dd* format.

This value is optional. If you omit this value, a blank is set in the **End Date** field at the Event Console.

EndTime

Specify the event end time in *hh:mm:ss* format.

This value is optional. If you omit this value, a blank is set in the **End Time** field at the Event Console.

ResultCode

Specify the event exit code as a number or as a variable that stores this value.

This value is optional. If you omit this value, a blank is set in the **Result Code** field at the Event Console.

Description

The `IMEventMessage` command issues an event to JP1/IM or JP1/Base according to the specified parameters. The command returns `True` on successful execution, or `False` if an error occurs.

Events issued using this command have `00003B03` as their event ID. The **Product Name** is set to `/HITACHI/JP1/SCRIPT`, and the **Object Type** is set to `PROCESS`.

Supplement

In JP1/IM, the extended attribute description file for the event console is stored in the `DATA` folder under the installation folder. Its file name is `SCRIPT_BASE.en`.

Important note

JP1/Script can issue events only to the physical host on which JP1/Base is running, not to a logical host. Therefore, if you are using JP1/IM in a logical host environment, you will not be able to view JP1 events issued by JP1/Script unless you enable transfer such events by entering the appropriate setting in the forwarding setting file at the JP1/Base physical host.

Example

```
Dim se_Date, s_Time, e_Time, rtn1
se_Date = Date()
s_Time = Time()
```

```
rtn1 = Exec (_SCF_+"ABC.EXE", True)
e_Time = Time()

If rtn1 Then
    IMEventMessage ("ABC.EXE ended normally.", , _USER_, _
        _SCF_FIL_, End, se_Date, s_Time, se_Date, _
        e_Time, _EXEC_RTN_)
Else
    IMEventMessage ("ABC.EXE terminated with an error.", , _
        _USER_, _SCF_FIL_, End, se_Date, s_Time, _
        se_Date, e_Time, _RTN_)
End
```

JP1/Script version

Supported from JP1/Script 05-20.

8.7 Command for menu display

8.7.1 Menu (display a user-defined menu form)

Purpose

Displays a user-defined window (menu form).

Syntax

```
Menu (FormName, [SpnFilePath] [, FocusField])
```

Arguments

FormName

Specify the form name of the menu as a character string or as a variable that stores this value.

SpnFilePath

Specify the full path of the menu information file that contains information about the menu to be displayed. Specify the path as a character string or as a variable that stores this value.

This value is optional. If you omit this value, the menu information file for the executing script is assumed. If the script was invoked by the `CallSpt` command, the menu information file for the called script file is assumed.

FocusField

Specify the field name that has the focus at menu startup. Write the field name as a character string or as a variable that stores this value.

This value is optional. If you omit this value, the field defined in the menu information file has the focus.

Description

The `Menu` command displays a user-defined menu form. The command returns `True` on successful execution, or `False` if an error occurs.

When the menu form ends, one of the following values indicating the reason for termination is stored in the `_FORM_TERM_KEY_` reserved variable:

Value	Meaning
Enter	The user pressed the Enter key.
Esc	The user pressed the Esc key.
F1 to F12	The user pressed a function key (F1 to F12).
Command	The user executed a command.
Close	The user pressed Alt + F4 or the Close button.
Timeout	The timeout was exceeded.

When the reason for termination is `Command`, the number of the executed command is stored in the `_FORM_TERM_CMDNO_` reserved variable.

One of the following strings representing the modifier key of the termination key is set in the `_FORM_MODIFY_KEY_` reserved variable:

Value	Meaning
Shift	Shift key

Value	Meaning
Ctrl	Ctrl key
Alt	Alt key
Shift_Ctrl	Shift key and Ctrl key
Ctrl_Alt	Ctrl key and Alt key
Shift_Alt	Shift key and Alt key
Shift_Ctrl_Alt	Shift key and Ctrl key and Alt key

In addition, the name of the field that last had the focus is set in the `_FORM_FIELD_NAME_` reserved variable.

Note

You cannot use this command in a script started as a service. An execution error occurs if this command is used. If the external command executed from the `Menu` command returns a nonzero value as the termination code, a message box is displayed. To hide this message box, set 0 in the following registry key:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1/Script\SPTX

Value name

Menu_EmgMsgBox

Value datatype

REG_DWORD

Value

- 0: Hide the message box.
- 1: Display the message box

When the setting takes effect

The setting takes effect the next time the script file is executed.

Example

```
' Cancel script execution if the menu terminates because
' the Esc key was pressed.
Menu ("MyMenu")
If _FORM_TERM_KEY_ = "Esc" Then
    Exit
Else
    ...
    ...
    ...
End
```

JP1/Script version

Supported from JP1/Script 05-20.

8.8 Commands for performing calculations

8.8.1 + operator (addition) (find the sum of two expressions)

Purpose

Finds the sum of two expressions.

Syntax

```
Result = Expression1 + Expression2
```

Arguments

Result

Specify a variable for storing the sum of the two expressions.

Expression1

Write any expression.

Expression2

Write any expression.

Description

The + operator adds *Expression1* and *Expression2* and returns the sum.

The action of the + operator depends on the internal processing of the two expressions, as follows.

Condition	Operation performed
Both expressions are strings.	String concatenation
Both expressions are numbers.	Addition
Both expressions are strings consisting entirely of numbers.	Addition
One expression is a number, and the other is a string.	String concatenation
One expression is a string, and the other is a string consisting entirely of numbers.	String concatenation
One expression is a number, and the other is a string consisting entirely of numbers.	Addition

If both expressions are Empty values, the number 0 is set in *Result*. However, if only one of the expressions is an Empty value, the other expression is returned as is in *Result*.

Note

When *Result* and *Expression1* are the same variable, you can use the += operator instead.

Examples

```
' This codes stores "ABCDEF" in variable result1.
result1 = "ABC" + "DEF"

' This codes stores 12 in variable result2.
result2 = 7 + 5

' This codes stores "10 min." in variable result3.
result3 = 10 + "min."
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.2 += operator (addition) (assign to a variable the sum of a variable and an expression)

Purpose

Adds an expression to a variable, and assigns the result to the variable.

Syntax

```
Result += Expression
```

Arguments

Result

Specify a variable for storing the addition result.

Expression

Write any expression.

Description

The += operator adds the value of an expression to the value of a variable, and assigns the sum to the variable.

The action of the += operator depends on the same conditions as the + operator, as follows.

Condition	Operation performed
Both expressions are strings.	String concatenation
Both expressions are numbers.	Addition
Both expressions are strings consisting entirely of numbers.	Addition
One expression is a number, and the other is a string.	String concatenation
One expression is a string, and the other is a string consisting entirely of numbers.	String concatenation
One expression is a number, and the other is a string consisting entirely of numbers.	Addition

If both expressions are Empty values, the number 0 is set in *Result*. If *Result* is an Empty value or undefined, the value of *Expression* is assigned to *Result*. If *Expression* is an Empty value, *Result* remains unchanged.

Examples

```
' This code stores "ABCDEF" in variable result1.
result1 = "ABC"
result1 += "DEF"

' This code stores 11 in variable result2.
result2 = 10
result2 += 1

' This code stores 1 in variable result3.
Dim result3
result3 += 1
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.3 - operator (subtraction and negation) (find the difference between two numbers)

Purpose

Finds the difference between two numbers, or reverses the sign of a numerical expression.

Syntax 1

```
Result = Number1 - Number2
```

Syntax 2

```
-Number
```

Arguments

Result

Specify a variable for storing the difference between the two numbers.

Number

Write any numerical expression.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

In Syntax 1, the - operator subtracts *Number2* from *Number1* and returns the difference as the result.

In Syntax 2, the - operator is the unary negation operator and reverses the sign of an expression.

Expressions with Empty values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the -= operator instead.

Example

```
' This code stores 2 in result1.  
result1 = 7 - 5
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.4 -= operator (subtraction) (assign to a variable the difference between a variable and an expression)

Purpose

Subtracts an expression from a variable, and assigns the result to the variable.

Syntax

```
Result -= Number
```

Arguments

Result

Specify a variable for storing the subtraction result.

Number

Write any numerical expression.

Description

The `--` operator subtracts the value of the expression *Number* from the value of the variable *Result*, and assigns the difference to the variable *Result*.

If both values are `Empty`, the number 0 is set in *Result*. If *Result* is an `Empty` value or undefined, `-Number` (the numerical expression with its sign reversed) is assigned to *Result*. If *Number* is an `Empty` value, the value of *Result* is unchanged.

Examples

```
' This code stores -1 in variable result1.
Dim result1
result1 -= 1

' This code stores 9 in variable result2.
result2 = 10
result2 -= 1
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.5 Mod operator (remainder calculation) (find the division remainder of two numbers)

Purpose

Divides one number by another and returns the remainder.

Syntax

```
Result = Number1 Mod Number2
```

Arguments

Result

Specify a variable for storing the division remainder.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

The `Mod` operator divides *Number1* by *Number2*, and returns the remainder as the result.

Expressions with `Empty` values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the `Mod=` operator instead.

Example

```
' This code stores 5 in variable result1.  
result1 = 19 Mod 7
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.6 Mod= operator (remainder calculation) (assign to a variable the division remainder of a variable and an expression)

Purpose

Divides a variable by an expression, and assigns the remainder to the variable.

Syntax

```
Result Mod= Number
```

Arguments

Result

Specify a variable for storing the division remainder.

Number

Write any numerical expression.

Description

The Mod= operator divides the value of the variable *Result* by the value of the numerical expression *Number*, and assigns the remainder to *Result*.

Empty values are handled as zero. If *Result* is an Empty value or undefined, 0 is assigned to *Result*. If *Number* is an Empty value, an execution error occurs because this would be division by zero.

Example

```
' This code stores 5 in variable result1.  
result1 = 19  
result1 Mod= 7
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.7 * operator (multiplication) (find the product of two numbers)

Purpose

Finds the product of two numbers.

Syntax

```
Result = Number1 * Number2
```

Arguments

Result

Specify a variable for storing the product of the two numbers.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

The * operator multiplies *Number1* by *Number2* and returns the product.

Expressions with Empty values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the *= operator instead.

Example

```
' This code stores 35 in variable result1.  
result1 = 7 * 5
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.8 *= operator (multiplication) (assign to a variable the product of a variable and an expression)

Purpose

Multiplies a variable by an expression, and assigns the result to the variable.

Syntax

```
Result *= Number
```

Arguments

Result

Specify a variable for storing the multiplication result.

Number

Write any numerical expression.

Description

The *= operator multiplies the value of the variable *Result* by the value of the expression *Number*, and assigns the product to the variable *Result*.

Empty values are handled as zero. If *Result* is an Empty value or undefined, or if *Number* is an Empty value, 0 is assigned to *Result*.

Example

```
' This code stores 35 in variable result1.  
result1 = 7  
result1 *= 5
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.9 / operator (division) (find the quotient of two numbers)

Purpose

Finds the quotient of two numbers and returns an integer (same as the \ operator).

Syntax

```
Result = Number1 / Number2
```

Arguments

Result

Specify a variable for storing the quotient of the two numbers.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

The / operator divides *Number1* by *Number2*, and returns the quotient as the result.

Expressions with `Empty` values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the /= operator instead.

Example

```
' This code stores 2 in variables result1 and result2.
result1 = 14 / 7
result2 = 19 / 7
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.10 /= operator (division) (assign to a variable the quotient of a variable and an expression)

Purpose

Divides a variable by an expression, and assigns the integer part of the quotient to the variable (same as the \= operator).

Syntax

```
Result /= Number
```

Arguments

Result

Specify a variable for storing the quotient.

Number

Write any numerical expression.

Description

The `/=` operator divides the value of the variable *Result* by the value of the numerical expression *Number*, and assigns the quotient to *Result*.

Empty values are handled as zero. If *Result* is an Empty value or undefined, 0 is assigned to *Result*. If *Number* is an Empty value, an execution error occurs because this would be division by zero.

Example

```
' This code stores 2 in variables result1 and result2.
Dim result1, result2
result1 = 14
result2 = 19
result1 /= 7
result2 /= 7
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.11 \ operator (integer division) (find the quotient of two numbers)

Purpose

Finds the quotient of two numbers and returns an integer (same as the `/` operator).

Syntax

```
Result = Number1 \ Number2
```

Arguments

Result

Specify a variable for storing the quotient of the two numbers.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

The `\` operator divides *Number1* by *Number2*, and returns the quotient as the result.

Expressions with Empty values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the `\=` operator instead.

Example

```
' This code stores 2 in variables result1 and result2.
result1 = 14 \ 7
result2 = 19 \ 7
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.12 \= operator (integer division) (assign to a variable the quotient of a variable and an expression)

Purpose

Divides a variable by an expression, and assigns the integer part of the quotient to the variable (same as the /= operator).

Syntax

```
Result \= Number
```

Arguments

Result

Specify a variable for storing the quotient.

Number

Write any numerical expression.

Description

The \= operator divides the value of the variable *Result* by the value of the numerical expression *Number*, and assigns the quotient as an integer to *Result*.

Empty values are handled as zero. If *Result* is an Empty value or undefined, 0 is assigned to *Result*. If *Number* is an Empty value, an execution error occurs because this would be division by zero.

Example

```
' This code stores 2 in variables result1 and result2.
Dim result1, result2
result1 = 14
result2 = 19
result1 \= 7
result2 \= 7
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.13 ^ operator (power) (find the power of two numbers)

Purpose

Raises one number to the power of another number, and returns an integer.

Syntax

```
Result = Number1 ^ Number2
```

Arguments

Result

Specify a variable for storing the result.

Number1

Write any numerical expression.

Number2

Write any numerical expression.

Description

The `^` operator raises the value of *Number1* by the power of *Number2*, and returns an integer result. Expressions with `Empty` values are handled as zero.

Note

When *Result* and *Number1* are the same variable, you can use the `^=` operator instead.

Example

```
' This code stores 8 in variable result1.  
result1 = 2 ^ 3
```

JP1/Script version

Supported from JP1/Script 06-00.

8.8.14 ^= operator (power) (assign to a variable a variable raised to the power of an expression)

Purpose

Raises a variable to the power of an expression, and assigns the result to the variable.

Syntax

```
Result ^= Number
```

Arguments

Result

Specify a variable for storing the result.

Number

Write any numerical expression.

Description

The `^=` operator raises the value of the variable *Result* by the power of the numerical expression *Number*, and assigns the result as an integer to *Result*.

`Empty` values are handled as zero. If *Result* is an `Empty` value or undefined, 0 is assigned to *Result*. If *Number* is an `Empty` value, the result is always 1 because the variable would be raised to the power of zero.

Example

```
' This code stores 8 in variable result1.  
result1 = 2  
result1 ^= 3
```

JP1/Script version

Supported from JP1/Script 06-51.

8.8.15 Comparison operators (=, <>, <, <=, >, >=) (compare two expressions)

Purpose

Compares two expressions.

Syntax

```
Result = Expression1 comparison-operator Expression2
```

Arguments

Result

Specify a variable for storing the comparison result.

Expression1

Write any expression.

Expression2

Write any expression.

comparison-operator

Specify a comparison operator.

Description

The comparison operators compare two expressions. Each operator returns a `True` or `False` result according to the following conditions:

= (equal)

True condition: *Expression1* = *Expression2*

False condition: *Expression1* <> *Expression2*

<> (not equal)

True condition: *Expression1* <> *Expression2*

False condition: *Expression1* = *Expression2*

< (less than)

True condition: *Expression1* < *Expression2*

False condition: *Expression1* >= *Expression2*

<= (less than or equal)

True condition: *Expression1* <= *Expression2*

False condition: *Expression1* > *Expression2*

> (greater than)

True condition: *Expression1* > *Expression2*

False condition: *Expression1* <= *Expression2*

>= (greater than or equal)

True condition: *Expression1* >= *Expression2*

False condition: *Expression1* < *Expression2*

The datatypes of the expressions determine the type of comparison performed and the result, as follows:

Condition	Operation performed
Both expressions are numeric datatypes.	Numeric comparison
Both expressions are string datatypes.	String comparison [#]
One expression is a numeric datatype, and the other is a string datatype.	String comparison [#]
One expression is an Empty value, and the other is a numeric datatype.	Numeric comparison with the Empty value taken as zero except in equality comparisons.
One expression is an Empty value, and the other is a string datatype.	String comparison with the Empty value taken as a zero-length string (""). [#]
Both expressions are Empty values.	The two expressions are considered to be equal.

#

String comparison is performed in dictionary order.
Note that one-byte characters are not case sensitive.

JP1/Script version

Supported from JP1/Script 01-00.

8.8.16 And operator (logical AND) (find the logical AND of two expressions)

Purpose

Finds the logical AND of two expressions.

Syntax

```
Result = Expression1 And Expression2
```

Arguments

Result

Specify a variable for storing the result.

Expression1

Write any expression that evaluates to true or false.

Expression2

Write any expression that evaluates to true or false.

Description

The And operator finds the logical AND of two expressions. The *Result* value is given as follows:

Expression1	Expression2	Result
True	True	True
True	False	False
False	True	False
False	False	False

Example

```
' If the value of variable number1 is between 1 and 10,  
' invoke executable file "Day10.SPT".  
number1 = Day()  
If 1 <= number1 And number1 <= 10 Then  
    Exec ("Day10.SPT", True)  
End If
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.17 Or operator (logical OR) (find the logical OR of two expressions)

Purpose

Finds the logical OR of two expressions.

Syntax

```
Result = Expression1 Or Expression2
```

Arguments

Result

Specify a variable for storing the result.

Expression1

Write any expression that evaluates to true or false.

Expression2

Write any expression that evaluates to true or false.

Description

The Or operator finds the logical OR of two expressions. The *Result* value is given as follows:

Expression1	Expression2	Result
True	True	True
True	False	True
False	True	True
False	False	False

Example

```
' If the value of variable number1 is 7 or less, or 20 or  
' greater, invoke executable file "Backup.BAT".  
number1 = Hour()  
If number1 <= 7 Or 20 <= number1 Then  
    Exec ("Backup.BAT", True)  
End If
```

JP1/Script version

Supported from JP1/Script 01-00.

8.8.18 Not operator (logical NOT) (find the logical NOT of an expression)

Purpose

Finds the logical NOT of an expression.

Syntax

```
Result = Not Expression
```

Arguments

Result

Specify a variable for storing the result.

Expression

Write any expression that evaluates to true not false.

Description

The `Not` operator finds the logical NOT of an expression. The `Result` value is given as follows:

Expression	Result
True	False
False	True

Example

```
' If the value of variable number1 is 10 or less, invoke  
' the executable file "Day10.SPT".  
number1 = Day()  
If Not 10 < number1 Then  
    Exec ("Day10.SPT", True)  
End If
```

JP1/Script version

Supported from JP1/Script 06-00.

8.9 Commands for performing evaluations

8.9.1 IsEmpty (check whether a variable is an Empty value)

Purpose

Checks whether a variable is an Empty value and returns True or False.

Syntax

```
IsEmpty (VarName)
```

Argument

VarName

Specify a variable to evaluate.

Description

The IsEmpty command checks whether the specified variable is an Empty value. The command returns True if the value is Empty, or False if not.

False is always returned if you specify an undefined variable.

Example

```
' Check whether the value of the _ALLRIGHT_ reserved
' variable is Empty.
If IsEmpty(_ALLRIGHT_) Then
  MessageBox("Log in again using an account with _
             Administrator rights.")
End If
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.2 IsDefine or IsDef (check whether a variable is defined)

Purpose

Checks whether a variable is defined and returns True or False.

Syntax

```
IsDefine (VarName)
IsDef (VarName)
```

Argument

VarName

Specify a variable to evaluate.

Description

The IsDefine or IsDef command checks whether the specified variable is defined. The command returns True if the variable is defined, or False if not.

Example

```
' Branch to another process if location variable %1
' is specified.
If IsDefine (%1) Then
    Exec ("CallProc01.SPT", True, %1)
Else
    Exec ("CallProc02.SPT", True)
End If
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.3 IsNumeric (check whether a value is a numeric)

Purpose

Checks whether a value can be evaluated as a numeric and returns `True` or `False`.

Syntax

```
IsNumeric (CheckValue)
```

Argument

CheckValue

Specify a value to evaluate or a variable that stores this value.

Description

The `IsNumeric` command checks whether the specified value can be evaluated as a numeric. The command returns `True` if the value can be evaluated as a numeric, or `False` if not.

JP1/Script version

Supported from JP1/Script 01-01.

8.9.4 IsEmptyDir (check whether a folder is empty)

Purpose

Checks whether a folder is empty and returns `True` or `False`.

Syntax

```
IsEmptyDir (DirName)
```

Argument

DirName

Specify the folder name as a string or as a variable that stores this value.

Description

The `IsEmptyDir` command checks whether the specified folder contains any files or folders. The command returns `True` if the folder is empty, or `False` if not. Note, however, that if you do not have access permissions to read the specified folder, the command returns `True` even if the folder is not empty.

`False` is always returned if you specify a non-existent folder.

Note

Take care when specifying the folder, or a folder in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Check whether folder "BKUP" is empty.
outDir = _SCF_+"BKUP\"
If ExistDir (outDir) = True Then
  If EmptyDir (outDir) = True Then
    MsgBox ("Empty !")
  Else
    MsgBox ("Not Empty!")
  End
Else
  MsgBox ("Not Exist!")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.5 IsExistDir (check whether a folder exists)

Purpose

Checks whether a folder exists and returns `True` or `False`.

Syntax

```
IsExistDir (DirName)
```

Argument

DirName

Specify the folder name as a string or as a variable that stores this value.

Description

The `IsExistDir` command checks whether a folder exists. The command returns `True` if the folder exists, or `False` if not. Note, however, that if you do not have access permissions to read the parent folder of the specified folder, the command returns `False` even if the folder exists.

Note

Take care when specifying the folder, or a folder in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Create a "SCRIPT" folder in the executable folder if one
' does not already exist.
path1 = _BIN_+"SCRIPT"
If ExistDir (path1) = False Then
  MakeDir (path1)
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.6 IsExistFile (check whether a file exists)

Purpose

Checks whether a file exists and returns `True` or `False`.

Syntax

```
IsExistFile (FilePath)
```

Argument

FilePath

Specify the full path of the file as a string or as a variable that stores this value.

Description

The `IsExistFile` command checks whether a file exists. The command returns `True` if the file exists, or `False` if not.

Supplement

If the specified folder name exists in the file to be checked, the command returns `True` as the execution result.

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

If you specify an existing folder name as the path of the file to be checked, the command returns `True` as the execution result.

Even if you do not have access permissions to read the existing folder specified by the path of the file being checked, the command returns `True` if the file exists.

Example

```
path1 = _BIN_+"SCRIPT\Logging.txt"
If IsExistFile (path1) = True Then
  Copy (path1, _TEMP_"ScpLog.txt")
  If _COPY_RTN_ <> Skip Then
    DeleteFile (path1)
  End
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.7 IsWritableDir (check whether a folder is writeable)

Purpose

Checks whether a folder is writeable and returns `True` or `False`.

Syntax

```
IsWriteableDir (DirName)
```

Argument

DirName

Specify the folder name as a string or as a variable that stores this value.

Description

The `IsWriteableDir` command checks whether the specified folder is writeable. The command returns `True` if the folder is writeable, or `False` if not.

`False` is always returned if you specify a non-existent folder.

Note

Take care when specifying the folder, or a folder in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Check whether the "BKUP" folder is writeable.
outDir = _SCF_+"BKUP\"
writeFlag = "NG"
If IsWriteableDir (OutDir) = True Then
    writeFlag ="OK"
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.8 IsFileAttribute or IsFileAttr (check a folder or file attribute)

Purpose

Checks whether a folder or file has a specific attribute and returns `True` or `False`.

Syntax

```
IsFileAttribute (PathName, Attribute)
IsFileAttr (PathName, Attribute)
```

Arguments

PathName

Specify the full path of the folder or file as a string or as a variable that stores this value.

Attribute

Specify one of the following attributes to check:

Value	Meaning
<code>ATTR_READONLY</code>	Read-only file
<code>ATTR_HIDDEN</code>	Hidden file
<code>ATTR_ARCHIVE</code>	Archive file
<code>ATTR_SUBDIR</code>	Sub-directory

Value	Meaning
ATTR_SYSTEM	System file
ATTR_TEMPORARY	Temporary file
ATTR_COMPRESSED (from version 05-20)	Compressed file

Description

The `IsFileAttribute` or `IsFileAttr` command checks whether the specified folder or file has a specific attribute. The command returns `True` if the folder or file has the attribute, or `False` if not.

To set folder or file attributes, use the `SetFileAttribute` command. To acquire the attributes of a folder or file, use the `GetFileAttribute` command.

Note

Take care when specifying the folder, or a folder or file in the folder set in the following environment variables: `ProgramFiles` (normally the Program Files folder on the system drive) and `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Delete a file if it has the temporary file attribute.
Dim file1
file1 = "C:\TEMP\logging.tmp"
If IsFileAttribute (file1, ATTR_TEMPORARY) = True Then
    DeleteFile (file1)
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.9 IsNew (compare files for the more recent version or date)

Purpose

Compares which of two files has the more recent version information or file date, and returns `True` or `False`.

Syntax

```
IsNew (PathName1, PathName2 [, Option])
```

Arguments

PathName1

Specify the full path of the first file as a string or as a variable that stores this value.

PathName2

Specify the full path of the second file as a string or as a variable that stores this value.

Option

Specify either of the following options:

Value	Meaning
Version	Compare the version information only.
FileTime	Compare the file dates only.

Description

If you omit the *Option* argument, the `IsNew` command compares the version information of the two files. If no version information is available, the command compares the file dates. `True` is returned if the file specified in *PathName1* is more recent than the file specified in *PathName2*. `False` is returned if the file specified in *PathName1* is older or the same version or date as the file specified in *PathName2*.

If you specify the *Option* argument, the files are compared accordingly.

If an error occurs, the command returns a zero-length string ("").

Note

Take care when specifying a file in the folder set in the environment variable `ProgramFiles` (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
' Compare the version information of file1 and file2.
' If file1 is more recent than file2, copy information from
' file2 to file1.
Dim file1, file2
file1 = _SCF_+"TEST.SPT"
file2 = "C:\BKUP\TEST.SPT"
If IsNew (file1, file2) = False Then
    Copy (file2, file1)
End
```

JP1/Script version

Supported from JP1/Script 05-10.

8.9.10 CheckDirName (check whether a folder name ends with a backslash)

Purpose

Checks whether a folder name ends with a backslash (\).

Syntax

```
CheckDirName (DirNameBuff [, Option])
```

Arguments

DirNameBuff

Specify the variable that stores the name of the folder to be checked.

Option

Specify the following optional value:

Value	Meaning
Remove	Remove the final backslash (\).

Description

If you omit the *Option* argument, the `CheckDirName` command checks the final character of the folder path stored in the specified variable, and appends a backslash (\) if the final character is not a backslash. If you specify `Remove` in *Option*, the `CheckDirName` command removes the final backslash.

The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
' Make sure the folder path ends with \, then create
' a full path.
Dim inDir , outDir
inDir = _TEMP_
outDir = "C:\Script\Data"
CheckDirName (inDir)
CheckDirName (outDir)
Copy (inDir+"FILE.DAT", outDir+"FILE.DAT", Overwrite)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.9.11 CheckDriveType (check the drive type)

Purpose

Checks the drive type.

Syntax

```
CheckDriveType (DrvTypeBuff, PathName)
```

Arguments

DrvTypeBuff

Specify a variable for storing the check result. The stored value will be one of the following:

Value	Meaning
REMOVABLE	Removable media (floppy disk drive, etc.)
FIXED	Hard disk
REMOTE	LAN or remote drive
CDROM	CD-ROM drive
RAMDISK	RAM disk
""	Unknown

PathName

Specify the drive path as a character string, or as a variable that stores this value.

Description

The `CheckDriveType` command checks the type of the drive in the specified path. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
Dim result1
CheckDriveType (result1, "D:\")
If result1 <> CDROM Then
    MsgBox ("The path is not a CD-ROM drive.", OK)
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.10 Commands for calling external programs

8.10.1 Exec (call an executable file on the local PC)

Purpose

Calls an executable file (EXE file, BAT file, COM file, SPT file, CMD file, or linked file) according to specified parameters. You can specify whether to wait for the called application to complete.

Linked files can be called in JP1/Script 05-00 and later versions.

Syntax

```
Exec (FileName, Flag [, Param1, Param2, ...])
```

Arguments

FileName

Specify the executable file as a character string or as a variable that stores this value.

You can specify any of the following types of files:

- Executable file (.EXE)
- MS-DOS batch file (.BAT)
- MS-DOS executable file (.COM)
- JP1/Script script file (.SPT)
- Command script (.CMD)
- Linked file

Flag

Specify `True` to wait for the specified executable file to complete; otherwise, specify `False`.

If you specified `False` in *Flag*, the script waits for the executable file to complete if it is still running when the script terminates. If you want to terminate the script without waiting for the executable file to complete, use the `Exit` command with `Skip` specified in *Option*. See Example 2.

Param1 to Param31

Specify the necessary parameters for executing the file specified in *FileName*. Write each parameter as a string or number or as a variable that stores this value.

In JP1/Script 06-00 and later versions, you can specify a one-dimensional array variable that stores all the required parameters.

For the parameter coding conventions, see [6.2 Rules for writing command lines](#).

If you specify one of the strings listed below as a parameter, the string is treated as the default when windows are displayed for an application executed by the `Exec` command. However, these parameters are not applicable to applications that control whether to hide, minimize, or maximize windows.

Parameter	Meaning
<code>/SPT:HIDE</code>	Hides the application window.
<code>/SPT:MIN</code>	Minimizes the application window to an icon.
<code>/SPT:MAX</code>	Maximizes the application window.

The parameters you specify in *Param1* to *Param31* are passed as command line parameters to the executable file. The strings `/SPT:HIDE`, `/SPT:MIN`, and `/SPT:MAX` are not passed as command line parameters.

Description

The `Exec` command executes a specified executable file.

If you specify `True` in *Flag*, the script waits for the called application to complete. If you specify `False`, the next command is processed without waiting for the application to complete.

The command returns `True` on successful execution, or `False` if an error occurs.

If you specify `True` in *Flag* and the execution result is `True`, the executable file's exit code is stored as a signed numeric in the `_EXEC_RTN_` reserved variable. Nothing is stored in this reserved variable if you specify `False` in *Flag* or if the execution result is `False`.

If you specify `False` in *Flag*, the executable file ID is stored in the `_EXEC_ID_` reserved variable.

Notes

- Depending on the type of file called by the `Exec` command, the execution result may differ even with the same type of error. For example, the command returns `False` when a non-existent EXE file is called, but `True` when a non-existent SPT file is called.

To determine whether the `Exec` command was successful with *Flag* set to `True`, you must refer to the exit code stored in the `_EXEC_RTN_` reserved variable in addition to the execution result. The first example below illustrates the recommended procedure.

- When you start an executable file that requires administrator permissions, the Windows User Account Control dialog box might appear. For details, see [1.8.2 Command behavior](#).
- The work folder of the executable file called by the `Exec` command is the same as the work folder of the script file.
- If double quotation marks (") are used in parameters, the values of the location variables that are passed to the executable file depend on whether the quotation marks are preceded by a space. See Example 3.

Example 1

```
' Call the script file "ABC.SPT".
rtn = Exec (_SCF_"ABC.SPT", True)
rtnCode = _RTN_

' Determine the execution result.
If rtn = True Then
  If _EXEC_RTN_ = 0 Then
    MsgBox ("Execution was successful.")
  Else
    MsgBox ("The script terminated abnormally. _
           Exit code="_EXEC_RTN_)
    Exit (_EXEC_RTN_)
  End
Else
  MsgBox ("The script terminated abnormally. _
         Error code="+rtnCode)
  Exit (rtnCode)
End
```

Example 2

```
' Call "ABC.EXE" but do not wait for its termination.
rtn1 = Exec ("ABC.EXE", False)
...
' Terminate the script without waiting for ABC.EXE
' to terminate.
Exit (0, Skip)
```

Example 3

1. If the double quotation marks (") are not preceded by a space:

Parameter format

```
Exec ( "C:\Temp\Test.SPT", True, "ABC""XYZ"" " )
```

Values set in the location variables

```
%0: C:\Temp\Test.SPT
```

```
%1: ABCXYZ
```

2. If the double quotation marks (") are preceded by a space:

Parameter format (Δ : space)

```
Exec ( "C:\Temp\Test.SPT", True, "ABC Δ ""XYZ"" " )
```

Values set in the location variables

```
%0: C:\Temp\Test.SPT
```

```
%1: ABC
```

```
%2: XYZ
```

JP1/Script version

Supported from JP1/Script 01-00.

8.10.2 NetExec (call an executable file on the local PC or remote PC)

Purpose

Calls an executable file (EXE file, BAT file, COM file, SPT file, CMD file, or linked file) on the local PC, or on a remote PC, according to specified parameters. You can specify whether to wait for the called application to complete.

Syntax

```
NetExec ([CompName], FileName, Flag, [ExecDirName], [ExecPlace], [Option]  
[, Param1, Param2, ...])
```

Arguments

CompName

Specify the computer at which to call the executable file, using a character string or a variable that stores this value.#

This value is optional. If you omit this value, the computer on which the command is executed is assumed.

JP1/Script must be installed on the remote computers you want to specify. For details about accounts for accessing computers, see [2.2 Accounts for communication with other computers](#).

Note: Specify the following values if the target computer is in a cluster system environment:

- If the JP1/Script service, Script Launcher, or Script Launcher service has been registered as a resource, specify a logical host name or a logical IP address.
- If the JP1/Script service, Script Launcher, or Script Launcher service has not been registered as a resource, specify a physical host name or a physical IP address.

FileName

Specify the executable file as a character string or as a variable that stores this value.

You can specify any of the following types of files:

- Executable file (.EXE)

- MS-DOS batch file (.BAT)
- MS-DOS executable file (.COM)
- JP1/Script script file (.SPT)
- Command script (.CMD)
- Linked file

When you specify a path or file name containing a space, there is no need to enclose the entire path or file name in double quotation marks ("").

To specify only a file name, specify the folder containing the target executable file for the PATH environment variable of the computer specified in *CompName*.

Flag

Specify `True` to wait for the specified executable file to complete; otherwise, specify `False`.

If you specified `False` in *Flag*, the script waits for the executable file to complete if it is still running when the script terminates. If you want to terminate the script without waiting for the executable file to complete, use the `Exit` command with `Skip` specified in *Option*. See *Example 2* in *8.10.1 Exec (call an executable file on the local PC)*.

ExecDirName

Specify the current directory containing the executable file specified in *FileName*, expressed as a character string or the name of the variable that contains the applicable value.

This value is optional. When it is omitted, the command assumes the directory of the executable file.

This value is applicable if the executable file specified in *FileName* is not a JP1/Script file.

ExecPlace

Specify `True` to execute the executable file specified in *FileName* in the service space, or `False` to execute it in the logon space.

This value is optional. If you omit this value, `True` is assumed.

You can specify `True` only if the JP1/Script service is active on the computer specified in *CompName*. If you specify `False`, Script Launcher or Script Launcher service must be active on the specified computer.

Option

If you specified a JP1/Script script file (.SPT) in *FileName*, you can specify the following optional value:

Value	Meaning
CopyFile	<p>Copies the script file from the calling side to the computer specified in <i>CompName</i>.</p> <p>At the destination computer, a folder with the name of the calling computer is created in the DATA folder# in the installation folder. The script file is copied to this folder. The script's execution environment file (.SPV) is not copied. If the destination folder contains an execution environment file that has the same name as the script file, that execution environment file is deleted when the script file is copied.</p> <p>In JP1/Script 06-00 and later versions, if the script file has an associated menu information file (.SPN) of the same file name, that menu information file is copied with the script file. When these files are copied, any file that has the same name as these files is deleted from the destination folder.</p> <p>When the <code>NetExec</code> command terminated, the copied script file and menu information file are deleted. The analysis trace file and execution trace file which were output after execution of the copied script are copied back to the script execution folder on the calling side.</p> <p>Note the following if you specify <code>CopyFile</code> in <i>Option</i>.</p> <ul style="list-style-type: none"> • Any information specified in the execution environment file will be invalid. Therefore, the command lines specified in the execution environment file are invalid in the script file started by the <code>NetExec</code> command. • You cannot start multiple script files by using the <code>NetExec</code> command with <code>CopyFile</code> specified. • Regardless of the <i>Flag</i> setting, if you specify <code>CopyFile</code> in <i>Option</i>, do not log off Windows or shut down the system until the copied script file completes execution.

Value	Meaning
CopyFile	# The folder is created in the script execution environment folder (<i>system-drive</i> \ProgramData\Hitachi\Script\Data).

This value is optional. If you omit this value, the `Exec` command runs the script file residing on the computer specified in *CompName*.

This value is invalid if you specify an executable file other than a JP1/Script script file (`.SPT`) in *FileName*.

Param1 to *Param31*

Specify the necessary parameters for executing the file specified in *FileName*. Write each parameter as a string or number, or as a variable that stores this value.

In JP1/Script 06-00 and later versions, you can specify a one-dimensional array variable that stores all the required parameters.

For the parameter coding conventions, see [6.2 Rules for writing command lines](#).

If the strings listed below are specified in the parameters, they are treated as the defaults when the window is displayed for the application being started. These parameters are not applicable to an application that displays its window explicitly.

Parameter	Meaning
<code>/SPT:HIDE</code>	Hides the application window.
<code>/SPT:MIN</code>	Minimizes the application window to an icon.
<code>/SPT:MAX</code>	Maximizes the application window.

The parameters you specify in *Param1* to *Param31* are passed as command line parameters to the executable file. The strings `/SPT:HIDE`, `/SPT:MIN`, and `/SPT:MAX` are not passed as command line parameters.

Description

The `NetExec` command executes an executable file on a specified computer.

If you specify `True` in *Flag*, the script waits for the called application to complete. If you specify `False`, the next command is processed without waiting for the application to complete.

The command returns `True` on successful execution, or `False` if an error occurs.

If you specify `True` in *Flag* and the execution result is `True`, the executable file's exit code is stored as a signed numeric in the `_EXEC_RTN_` reserved variable. Nothing is stored in this reserved variable if you specify `False` in *Flag* or if the execution result is `False`.

If you specify `False` in *Flag*, the executable file ID is stored in the `_EXEC_ID_` reserved variable.

Note 1

Depending on the type of executable file called by the `NetExec` command, the execution result might differ even with the same type of error. For example, the command returns `False` if a non-existent EXE file is called, but returns `True` if a non-existent SPT file is called.

To determine whether the `NetExec` command with `True` specified in *Flag* terminated successfully, you must also refer to the exit code of the executable file stored in the `_EXEC_RTN_` reserved variable in addition to the execution result. For the recommended determination method, see [Example 1](#) in [8.10.1 Exec \(call an executable file on the local PC\)](#).

When a remote computer is specified in the *CompName* argument, operation is guaranteed only in a LAN environment. Operation in other environments such as a WAN environment cannot be guaranteed. Operation is not possible via a firewall.

Beginning with JP1/Script 07-01, if an error occurs during NetExec execution, an error log file is output to the STXNetExec_Client or STXNetExec_Server folder, which is created under the LOG folder of the installation folder.

To suppress the output of error log files, specify 0 in the following registry:

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\
Option

Value name

Net_Trace

Value data type

REG_DWORD

Value

- 0: Do not output error log files.
- 1: Output error log files (default value).

When the setting takes effect

The setting takes effect the next time the script file is executed.

To change the output destination folder for error log files, specify the new folder name in the value of the following registry:

Value name

Net_Trace_Directory

Value data type

REG_SZ

Value

Output destination folder for error log files (initial value: script log folder)

When the setting takes effect

The setting takes effect the next time the script file is executed.

Note 2

When you start an executable file that requires administrator permissions, the Windows User Account Control dialog box might appear. For details, see *1.8.2 Command behavior*.

Executable files called by the NetExec command do not inherit the process environment variable of the calling process. Therefore, the values set on the calling side cannot be referenced.

Note 3

If you specify a non-existent computer name in *CompName* or specify a non-existent executable file name in *FileName*, the command returns True, and 0 is stored in the `_RTN_` reserved variable.

This also occurs in the following cases:

- Access to the executable file failed.
- A network error occurred.
- The JP1/Script service has stopped.
- Script Launcher or the Script Launcher service is not running.

Note 4

In JP1/AJS, programs that display a GUI to wait for the user's input cannot be directly executed as jobs. However, you can execute a program with a GUI in JP1/AJS by using a JP1/Script script file as follows:

1. Create a script file that executes the `NetExec` command (with `False` specified for *ExecPlace*) to call a program with a GUI.
2. Use a JP1/AJS job to execute the above script file.

Example

```
' Execute a script file residing on the computer
' "SOP4A065 (SCRIPT)" in the logon space.
comp1="SOP4A065 (SCRIPT) "
prm1="/SPALV(2) "
prm2="/SPXLV(2) "
rtn1=NetExec (comp1,_TEMP_"NETWORK.SPT", True, , False,
_, prm1, prm2)
' Terminate the script.
If rtn1 = True Then
' If the NetExec command terminates normally, return
' the exit code of the called script file.
Exit (_EXEC_RTN_)
Else
' If the NetExec command terminates abnormally, return
' the error detail code.
Exit (_RTN_)
End
```

JP1/Script version

Supported from JP1/Script 05-00.

8.10.3 WaitForExec (wait for completion or forcibly terminate an executable file)

Purpose

Waits for completion of or forcibly terminates an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`).

Syntax

```
WaitForExec ([Option], [ExecId] [, Time])
```

Arguments

Option

Specify one of the following optional values:

Value	Meaning
Wait	Wait for completion of a specific executable file called by the <code>Exec</code> or <code>NetExec</code> command.
WaitAll	Wait for completion of all executable files called by the <code>Exec</code> or <code>NetExec</code> command.
Abort	Forcibly terminate a specific executable file called by the <code>Exec</code> or <code>NetExec</code> command.
AbortAll	Forcibly terminate all executable files called by the <code>Exec</code> or <code>NetExec</code> command.

This value is optional. If you omit this value, `Wait` is assumed.

ExecId

If you specified `Wait` or `Abort` in *Option*, specify the ID of an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`). Write the file ID as a character string or as a variable that stores the value.

This value is optional. If you omit this value, the value stored in the `_EXEC_ID_` reserved variable is assumed.

Time (from version 06-00)

Specify a wait interval in seconds. Specify a number or zero, or a variable that stores this value.

If you specified `Wait` or `WaitAll` in *Option*, the script waits during the specified interval for the executable file(s) to terminate.

If you specified `Abort` or `AbortAll` in *Option*, the executable file(s) are forcibly terminated when the wait interval has elapsed.

This argument is optional. Omit if not required.

Regardless of the *Option* specification, the `WaitForExec` command returns `True` if the executable file(s) terminated by or at the specified time, or `False` if the executable file(s) did not terminate. The command also sets the same value as the `_ERR_TIMEOUT_` reserved variable in the `_RTN_` reserved variable. The execution control does not jump to the label specified in the `On Error` statement.

Description

The `WaitForExec` command waits for completion of or forcibly terminates an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`). The command returns `True` on successful execution, or `False` if an error occurs.

`True` is always returned if the executable file has already completed when this command is invoked.

Example

```
Dim exec1_ID, exec2_ID
If Exec (_WIN_+"NOTEPAD.EXE", False, _SCF_+"ABC.SPT") Then
    exec1_ID = _EXEC_ID_
    ...
End
If Exec (_WIN_+"NOTEPAD.EXE", False, _BIN_+"Memo.TXT") Then
    exec2_ID = _EXEC_ID_
    ...
End

' Wait for Notepad to complete.
WaitForExec (Wait, exec1_ID)
' Forcibly terminate Notepad after 60 seconds.
WaitForExec (Abort, exec2_ID, 60)
```

JP1/Script version

Supported from JP1/Script 01-01.

8.10.4 GetExecStatus (get the execution status of an executable file)

Purpose

Obtains the current execution status of an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`).

Syntax

```
GetExecStatus ([ExecId])
```

Argument

ExecId

Specify the ID of an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`). Write the file ID as a character string or as a variable that stores the ID value. This ID is stored in the `_EXEC_ID_` reserved variable at execution of the `Exec` or `NetExec` command.

This value is optional. If you omit this value, the value stored in the `_EXEC_ID_` reserved variable is assumed.

Description

The `GetExecStatus` command obtains the current execution status of an executable file called by the `Exec` or `NetExec` command (the *Flag* argument must be `False`). On successful execution, the command returns the execution status of the executable file. If an error occurs, a zero-length string ("") is returned.

Either of the following values is returned as the execution status:

Value	Meaning
<code>EXEC_RUNNING</code>	The executable file is executing.
<code>EXEC_STOPPED</code>	The executable file is not executing.

Example

```
' After termination of processing of the executable file
' called by the Exec command (Flag argument False), start
' processing of the next Exec command.
Dim execl_ID
prm1 = _SCF_"Result.TXT"
If Exec (_SCF_"Sum.SPT", False, prm1) Then
    execl_ID = _EXEC_ID_
    ...
End
While GetExecStatus (execl_ID) = EXEC_RUNNING
    Sleep (100)
End While
Exec (_WIN_"NOTEPAD.EXE", True, prm1)
```

JP1/Script version

Supported from JP1/Script 06-00.

8.10.5 CallSpt (call a script file with multiple parameters set)

Purpose

Calls a script file (.SPT file) with multiple parameters set. Unlike the `Exec` command, the called script file is executed within the current process.

Syntax

```
CallSpt (SptFileName [, Param1, Param2, ... ,Param31])
```


Arguments

SptFileName

Specify the name of the JP1/Script script file (`.SPT`) as a character string, or as a variable that stores this value. The extension `.SPT` is appended automatically if omitted.

Param1 to Param31

Specify the necessary parameters for executing the file specified in *SptFileName*. Write each parameter as a string or number, or as a variable that stores this value.

Alternatively, you can specify a one-dimensional array variable that stores all the required parameters. For parameter coding conventions, see [6.2.3 Command line coding conventions](#).

To include *n* double quotation marks (") as characters in a parameter, write $n*2$ double quotation marks, and enclose the entire parameter with double quotation marks.

If specified in the `CallSpt` command, the `/SPT:HIDE`, `/SPT:MIN`, and `/SPT:MAX` parameters, which have a special meaning in the `Exec` and `NetExec` commands, and the `/SPALV(n)`, `/SPXLV(n)`, and `/NOEVLOG` parameters are passed as command line parameters and handled as character strings. They have no special meaning in this command.

Description

The `CallSpt` command executes a specified script file.

Unlike the `Exec` command, a script file called by the `CallSpt` command is executed within the current process. The script file therefore behaves like a procedure, sharing the environment of the current process and able to update the process environment variable.

The following variable and procedure relationships apply between a script file that executes the `CallSpt` command (hereafter, *parent script*) and a script file specified in the *SptFileName* argument (hereafter, *child script*):

- Variables that are explicitly declared in a child script, using the `Dim` command or equivalent, behave as local variables that are valid only within that child script.
- Variables that are used but not explicitly declared in a child script are also local, unless they are defined in the parent script before the `CallSpt` command is invoked, or in a parent script at a level higher than the calling parent script. A variable defined in this way outside the child script is handled as a global variable.
- Procedures defined within the parent script, or in a higher-level parent script, can also be used within the child script.
- When procedures of the same name are defined in the parent script and child script, the procedure defined in the child script is valid in that script. However, if the two procedures are `Function` and `Sub`, respectively, the `Function` procedure defined in the parent script is valid in the child script.
- Local variables and procedures defined in the child script, and location variables for the child script, are discarded when the child script completes execution (at termination of the `CallSpt` command in the parent script).
- Variables and procedures defined in the parent script cannot be used when executing the child script as an independent process.

Child scripts are executed according to the parameters specified for the parent script. Any `#FileVersion` or `#Option` setting in the child script is ignored. The settings in the child script's execution environment file (`.SPV`) do not apply.

The command returns `True` on successful execution, or `False` if an error occurs. If the execution result is `True`, the script file's exit code is stored as a signed numeric in the `_EXEC_RTN_` reserved variable.

Notes

- Unlike other commands, the syntax of the script file specified in *SptFileName* is also examined when you perform a syntax check of the `CallSpt` command.

- Null characters specified in parameters are ignored. To omit a parameter, specify, for example, an explicit string that will be passed from the calling script to the called script. See Example 2.
- If a string containing spaces is specified for a parameter, the parameter is passed to the called script as a single string delimited by spaces. In this case, enclose the entire string containing spaces in double quotation marks ("). However, you need to remove the double quotation marks (") from the parameters received at the called script. See Example 3.

Important note

- The `Exit` command executed in a script called by the `CallSpt` command simply terminates execution of that script and returns control to the calling script. Execution of the calling script is not terminated.
- The `Skip` specification in the *Option* argument of the `Exit` command is valid if you terminate the script without waiting for completion of an executable file called by the `Exec` or `NetExec` command. In this case, to terminate the script from a script file called by the `CallSpt` command, specify `Skip` in the *Option* argument of the `Exit` command for a script file that executes the `CallSpt` command. The script will not terminate if you specify `Skip` in the *Option* argument of the `Exit` command for a script file that executes the `Exec` or `NetExec` command. For details, see *Example 4*.

Example 1

```
' Execute the script file "Environment.SPT", and reference
' and update the process environment variable.
Dim Path01
rtn = CallSpt(_SCF_"Environment.SPT", "Get", "Path01", _
             Path01)

...
rtn = CallSpt(_SCF_"Environment.SPT", "Set", "Path01", _
             TEMP_)

' Environment.SPT processing
Select Case %1
Case "Set"
    SetEnv(ProcessEnv, %2, %3)
Case "Get"
    %3 = GetEnv(ProcessEnv, %2)
End Select
```

Example 2

In the specification shown below, the `NULL` parameter (null character "") of the `CallSpt` command in the calling script is ignored.

The location variables, where "B" is stored in %1 and "D" is stored in %2, are passed to the called script.

Calling script

```
CallSpt("A.spt", "B", "", "D")
```

Called script

```
Parm1 = %1 ' "B"
Parm2 = %2 ' "D"
Parm3 = %3 ' null character""
```

To omit a parameter, specify, for example, an explicit string that will be passed from the calling script to the called script as shown below.

Calling script

```
Dim C_Parm
If ... Then
    C_Parm="C"
Else
    C_Parm="***NOTPARAMETA***"
End If
CallSpt("A.spt", "B", C_Parm, "D")
:
```

Called script

```
If %2 <> "***NOTPARAMETA***" Then
    Parm1 = %1 ' "B"
    Parm2 = %2 ' "C"
    Parm3 = %3 ' "D"
Else
    Parm1 = %1 ' "B"
    Parm2 = "" ' Null characters""
    Parm3 = %3 ' "D"
End If
```

The number of arguments used when the function was called is stored in the `_ARGV_CNT_` reserved variable. If you want to check whether any arguments are omitted, use `_ARGV_CNT_`.

Example 3

In the specification shown below, the location variables in the calling script are as follows: "My" is stored in %1, "Documents" is stored in %2, and "Files" is stored in %3.

```
CallSpt("A.spt", "My Documents", "Files")
```

To specify a parameter containing spaces, enclose the entire string in double quotation marks (") and pass it to the called script. Note that because the string containing double quotation marks is passed to the called script, you need to remove the double quotation marks from the parameter as shown below.

Calling script

```
CallSpt("A.spt", """"My Documents""", "Files")
:
```

Called script

```
Dim Parm1
Dim Path1
Parm1= Mid (%1, 2, Len (%1) - 2)
Path1 = Parm1 + "\" + %2
:
```

Example 4

Terminate the calling script without waiting for completion of the called executable file.

Calling script: SampleA.SPT

```
' Call the Script file "SampleB.SPT".
CallSpt(_SCF_"SampleB.SPT")
:
```

```
' Terminate the script without waiting for completion of  
' SampleC.EXE.  
Exit(0, Skip)
```

Called script: SampleB.SPT

```
' Call the executable file "SampleC.EXE" but do not wait  
' for its termination.  
Exec(_SCF_"SampleC.EXE", False)
```

JP1/Script version

Supported from JP1/Script 06-71.

8.11 Automatic start commands

8.11.1 EntryStartUp (register a script file for automatic startup)

Purpose

Registers a script file for automatic startup.

Syntax

```
EntryStartUp (FilePath, [MultiExec], [ExecType], [ExecTime], [ExecWeek],  
[ExecDay] [, TerminateTime])
```

Arguments

FilePath

Specify the full path of the script file as a character string, or as a variable that stores this value. The file extension `.SPT` is appended automatically if omitted.

MultiExec

Specify whether the script file can be activated concurrently from multiple processes. Write `True` to allow multi-activation, or `False` to prohibit multi-activation.

This value is optional. If you omit this value, the value set in the execution environment file is assumed.

ExecType

Specify the startup type as either of the following values:

Value	Meaning
Logon	Start the script at logon.
Service	Start the script as a service.

This value is optional. If you omit this value, the value set in the execution environment file is assumed.

ExecTime

If required, specify a start time in `hh:mm:ss` format. Always write `00` in `ss`. Any other value will be switched to `00`.

Omit this argument if you do not want to activate the script at a particular time.

ExecWeek

If required, specify a day of the week to activate the script. Specify the day as a number or as a variable that stores this value.

The value is interpreted as follows:

Value	Meaning
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday

Value	Meaning
7	Saturday

Omit this argument if you do not want to activate the script on a particular day of the week.

ExecDay

If required, specify a date on which to activate the script. Specify the date as a number in the range 1 to 31, or as a variable that stores this value.

Omit this argument if you do not want to activate the script on a particular date.

The *ExecDay* value is ignored if you set *ExecWeek*.

TerminateTime

Specify a time to terminate the script as a number or as a variable that stores this value. Specify a value in the range 0 to 1,440 (minutes).

This value is optional. If you omit this value, the value set in the execution environment file is assumed.

Description

The `EntryStartUp` command registers a script file for automatic startup. The command returns `True` on successful execution, or `False` if an error occurs.

This command updates the contents of the execution environment file (`.SPV`) associated with the script file specified in *FilePath*. If no such file exists, a new execution environment file is created.

Example

```
' Register for automatic startup only those script files
' in the D:\Work\Script folder that have an execution
' environment file.
wkPath = "D:\Work\Script\"
' Get file names with extension .SPT in FileName.
For FileName = wkPath + "*" + _SCF_EXT_ Do
  ' Concatenate the folder name and the file name, and set it
  ' in FileName.
  FileName = wkPath + FileName
  ' Change the file name extension to .SPV, and set the file
  ' names in the FilePath.
  FilePath = Left (FileName, Len(FileName) - 4) + _SVF_EXT_
  If IsExistFile (FilePath) = True Then
    EntryStartUp (FileName)
  End
End For
```

JP1/Script version

Supported from JP1/Script 05-10.

8.11.2 CancelStartUp (cancel automatic startup for a script file)

Purpose

Cancels automatic startup for a script file.

Syntax

```
CancelStartUp ([FilePath])
```

Argument

FilePath

Specify the full path of the script file as a character string, or as a variable that stores this value. The file extension `.SPT` is appended automatically if omitted.

This value is optional. If you omit this value, automatic startup is canceled for all the script files that have been registered to start automatically.

Description

The `CancelStartUp` command cancels automatic startup for a specified script file. The command returns `True` on successful execution, or `False` if an error occurs.

`True` is always returned if you specify a non-existent script file, or if there is no automatic start setting in the execution environment file for the specified script file.

Example

```
' Cancel automatic startup for all script files one hour  
' after this script file is activated.  
Sleep (60 * 60 * 1000)  
CancelStartUp ()
```

JP1/Script version

Supported from JP1/Script 05-10.

8.12 Comment command

8.12.1 Rem or ' (write a comment in a program)

Purpose

Used to write a comment in a program.

Syntax

```
Rem Comment  
' Comment
```

Argument

Comment

Write a comment to clarify the programming. You must enter one or more spaces between *Rem* and *Comment*.

Description

A single quotation mark (') can be used instead of *Rem*. The quotation mark or *Rem* can be written on the same line following a statement or command.

Example

```
Rem Output a message to a file.  
Message (Target_File, _SCF_+"Logging.TXT", "Execution  
started.")  
  
' Display a message in a window.  
Message (Target_Dispon, "Logging", "Execution started.")
```

JP1/Script version

Supported from JP1/Script 01-00.

8.13 Other commands

8.13.1 Sleep (halt script execution)

Purpose

Halts script execution for a specified time.

Syntax

```
Sleep (Time)
```

Argument

Time

Specify the sleep time as a numeric value (milliseconds) beginning from 0 or as a variable that stores this value.

Description

The `Sleep` command halts script execution for a specified time.

Example

```
If debug = True Then  
    Sleep (3000)  
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.13.2 Alert (display or clear user error icons)

Purpose

Displays or clears user error icons in the status bar of Trace Viewer.

Syntax

```
Alert ([Mode])
```

Argument

Mode

Specify the status of Trace Viewer as one of the following values:

Value	Meaning
UserErr (from version 05-10) or True	Display a user error icon.
CancelUserErr (from version 05-10) or False	Clear the user error icon and reset to the executing icon.
Normal (from version 05-10)	Reset to the terminated normally icon.

This value is optional. If you omit this value, `True` is assumed.

Description

The `Alert` command displays user error icons in the status bar of Trace Viewer. It also clears an alert and returns the status display to the executing icon or terminated normally icon.

Example

```
path1 = _BIN_+"Logging.txt"
Message (Target_File, path1, "Execution started.\n", 30, 100)
...
...
Message (Target_File, path1, "Execution completed.")
If IsExistFile (path1) Then
    Alert (True)
End
```

JP1/Script version

Supported from JP1/Script 01-00.

8.13.3 Beep (sound a beep from the speakers)

Purpose

Sounds a beep from the speakers for a specified time.

Syntax

```
Beep ([Frequency] [, Time])
```

Arguments

Frequency

Specify the beep frequency as a number (Hz) or as a variable that stores this value. You can specify a value in the range 37 to 32,767.

This value is optional. If you omit this value, 440 Hz is assumed.

Time

Specify the length of the beep in milliseconds as a numeric value beginning from 0 or as a variable that stores this value.

This value is optional. If you omit this value, 100 milliseconds is assumed.

Description

The Beep command sounds a beep from the speakers.

Important note

- Depending on the OS and hardware environment, the Beep command might not sound a beep, or might cause one of the following errors:
 - 0001: The function is not correct.
 - 0002: The system cannot find the specified file.
 - 0005: Access was denied. Review and, if necessary, correct file attributes or security settings.
 - 1722: The RPC server cannot be used.
- Before you attempt to use the Beep command, check the command behavior in the execution environment. If the command does not sound a beep or causes an error, take the following actions.
 - (1) Execute the IMEventMessage command to display a message on the JP1/IM event console.
Example: IMEventMessage ("An error occurred", Error)

(2) To execute the `Beep` command in the service space, you must first create a script that executes the `Beep` command. Then use the `NetExec` command with the logon space specified to call the script that executes the `Beep` command. Note, however, that this operation is possible only when a command sounds a beep in the execution environment for the logon space.

Example

```
Beep (300, 200)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.13.4 Exit (terminate script execution)

Purpose

Terminates script execution.

Syntax

```
Exit ([Code] [, Option])
```

Arguments

Code

Specify the exit code as a number or as a variable that stores this value.

If you omit this value, 0 is assumed.

Option

Specify either of the following values:

Value	Meaning
Skip	Terminate the script without waiting for completion of executable files called by the <code>Exec</code> command or <code>NetExec</code> command.
Abort or Stop	When the script ends, forcibly terminate all executable files called by the <code>Exec</code> command or <code>NetExec</code> command.

This value is optional. If you omit this value, the script waits for completion of all executable files called by the `Exec` command or `NetExec` command.

Description

The `Exit` command terminates script execution. For details about JP1/Script exit codes, see [6.1.12 JP1/Script exit codes](#).

Important note

- The `Exit` command executed in a script called by the `CallSpt` command simply terminates execution of that script and returns control to the calling script. Execution of the calling script is not terminated.
- The `Skip` specification in the *Option* argument of the `Exit` command is valid if you terminate the script without waiting for completion of an executable file called by the `Exec` or `NetExec` command. In this case, to terminate the script from a script file called by the `CallSpt` command, specify `Skip` in the *Option* argument of the `Exit` command for a script file that executes the `CallSpt` command.

The script will not terminate if you specify `Skip` in the *Option* argument of the `Exit` command for a script file that executes the `Exec` or `NetExec` command. For details, see *Example 4* in [8.10.5 CallSpt \(call a script file with multiple parameters set\)](#).

Example

```
MessageBox ("Script execution will be terminated.", OK)
Exit (0)
```

JP1/Script version

Supported from JP1/Script 01-00.

8.13.5 GetErrorMessage (get an error message)

Purpose

Returns the error message associated with a specified error detail code.

Syntax

```
GetErrorMessage ([Code])
```

Argument

Code

Specify the error detail code as a number or as a variable that stores this value.

This value is optional. If you omit this value, the value currently stored in the `_RTN_` reserved variable is assumed.

Description

The `GetErrorMessage` command returns the error message associated with a specified error detail code. The error message is truncated if it exceeds 1,024 bytes.

Example

```
If Exec (_WIN_+"NOTEPAD.EXE", True, _BIN_+"Logging.txt")_
Then
  Exit (_EXEC_RTN_)
Else
  Dim errMsg
  errMsg = GetErrorMessage (_RTN_)
  Message (Target_File, _BIN_+"ErrLog.txt", errMsg)
  Exit (1)
End
```

JP1/Script version

Supported from JP1/Script 01-01.

9

Special Commands

This chapter describes the special commands you can use when creating a script.

9.1 List of special commands

Special commands are provided for advanced script operations, including registry and service operations.

Table 9-1 lists the special commands that you can use when creating a script. The numbers in parentheses in the Command column indicate the product version in which the command is supported. Commands without a following number are supported from version 01-00.

Take extra care when handling non-public product information with commands for registry and service operations.

Table 9–1: List of special commands

Category	Command	Description
Registry operations	RegRead	Reads a value from the registry.
	RegWrite	Enters a value in the registry.
	RegDelete	Deletes a value from the registry.
	RegDeleteKey	Deletes a registry subkey.
Graphics display	BitmapShow (05-00)	Draws a bitmap.
	BitmapHide (05-00)	Erases a bitmap.
Evaluations	IsEmptyReg	Checks whether a registry subkey is empty and returns True or False.
	IsExistRegKey	Checks whether a registry subkey exists and returns True or False.
	IsExistService	Checks whether a service exists and returns True or False.
	IsEmptyGroup (05-20)	Checks whether shortcuts exist in a program group and returns True or False.
Service operations	ServiceSetValue#	Sets a service information object for service operation.
	ServiceGetValue	Gets values stored in a service information object.
	ServiceCreate	Registers a service.
	ServiceDelete	Deletes a service.
	ServiceStart	Starts a registered service.
	ServiceStop	Stops a registered service.
	ServicePause	Halts an active service.
	ServiceContinue	Resumes a halted service.
	ServiceChange	Changes a setting for a service.
	ServiceQuery	Gets the information set for an active service.
	ServiceRefer	Gets the current status of a service.
	ServiceControl	Sends a control command to a service.
	GetServiceName (05-10)	Gets the service name from the display name of a service.
External program calls	CallDll	Calls DLL file(s). Multiple parameters may be specified.
Shortcuts	MakeGroup (05-20)	Creates a program group.
	DeleteGroup (05-20)	Deletes a program group.

Category	Command	Description
Shortcuts	MakeShortcut (05-20)	Creates a shortcut.
	DeleteShortcut (05-20)	Deletes a shortcut.
Process monitoring	GetProcessCount (05-20)	Gets the number of activations of a specified process and the process IDs.
	GetProcessInfo (05-20)	Gets process information for a specified process ID.
	TerminateProcess (05-20)	Forcibly terminates a process specified by process ID.
Other	ExitWindows (05-10)	Terminates script execution and logs off or shuts down Windows.
	SetRetryMode (10-00)	Sets the lock error retry function.
	ResetRetryMode (10-00)	Cancel the lock error retry function.
	SetTrialOpenMode (10-00)	Sets the trial-open function.
	ResetTrialOpenMode (10-00)	Cancel the trial-open function.

#

Before using commands for service operations, use the `ServiceSetValue` command to specify the necessary service information.

9.2 Commands for registry operations

9.2.1 RegRead (read a value from the registry)

Purpose

Reads a value from the registry.

Syntax

```
RegRead(RegKey, SubKey, ValueBuff, [EntryName], [TypeBuff] [, Option])
```

Arguments

RegKey

Specify the registry key as one of the following:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

If you specify HKEY_LOCAL_MACHINE and specify *Software* for the first key of *SubKey*, the entry is redirected to HKEY_CURRENT_USER\Software\Classes\VirtualStore\Machine\Software. The redirected entry will be read preferentially.

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

ValueBuff

Specify a variable for storing the read value.

EntryName

Specify the entry as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a value without an entry name is assumed.

TypeBuff

Specify a variable for storing the datatype.

This variable stores one of the following values:

- REG_BINARY
- REG_DWORD
- REG_DWORD_BIG_ENDIAN
- REG_EXPAND_SZ
- REG_LINK
- REG_MULTI_SZ
- REG_NONE
- REG_RESOURCE_LIST
- REG_SZ

This value is optional.

Option

Specify one of the following optional values:

Value	Meaning
10	Return a decimal number if the value is of type REG_DWORD or REG_DWORD_BIG_ENDIAN.
16	Return a hexadecimal number if the value is of type REG_DWORD or REG_DWORD_BIG_ENDIAN.

This value is optional. If you omit this value, 16 is assumed.

Description

The RegRead command reads a value from the registry and stores it in the specified variable. The command returns True on successful execution, or False if an error occurs.

Important note

If the type of the registry that was read is REG_MULTI_SZ, only the value in the first line is stored in the variable.

If the registry value to be read contains 1,025 or more bytes, an error occurs and 0234 (More data is available.) is output. Do not attempt to read a registry value longer than 1,024 bytes.

Example

```
Dim DBflag
RegRead(HKEY_LOCAL_MACHINE
        , "Software\Hitachi\JP1\Script\Option" _
        , DBflag, "Debug")
If DBflag = 1 Then
    Exit
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.2.2 RegWrite (enter a value in the registry)

Purpose

Enter a value in the registry.

Syntax

```
RegWrite(RegKey, SubKey, [EntryName], [Value] [, Type])
```

Arguments

RegKey

Specify the registry key as one of the following:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

If you specify a registry key other than `HKEY_CURRENT_USER`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

EntryName

Specify the entry as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a value without an entry name is assumed.

Value

Specify the value to be written to the registry. Write a string or number, or a variable that stores this value.

This value is optional. If you omit this value, a zero-length string or 0 is assumed.

Type

Specify the type of the value to be written to the registry as one of the following:

- `REG_DWORD`
- `REG_DWORD_BIG_ENDIAN`
- `REG_EXPAND_SZ`
- `REG_SZ`

This value is optional. If you omit this value, `REG_SZ` is assumed.

Description

The `RegWrite` command writes a value to a specified registry entry. The command returns `True` on successful execution, or `False` if an error occurs. The entry is created if no such entry exists.

Important note

If the type of the value to be written to the registry is `REG_SZ` or `REG_EXPAND_SZ`, a string consisting of no more than 1,024 bytes can be entered for the value. If you specify a string consisting of 1,025 or more bytes, the 1,025th and subsequent bytes are ignored.

Example

```
RegWrite(HKEY_CURRENT_USER
        , "Software\Hitachi\Script", "CurrentVersion"
        , "0100", REG_SZ)
```

JP1/Script version

Supported from JP1/Script 01-00.

9.2.3 RegDelete (delete a value from the registry)

Purpose

Deletes a value from the registry.

Syntax

```
RegDelete(RegKey, SubKey, EntryName)
```

Arguments

RegKey

Specify the registry key as one of the following:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

If you specify a registry key other than HKEY_CURRENT_USER, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

EntryName

Specify the entry as a character string or as a variable that stores this value.

This value is optional. If you omit this value, a value without an entry name is assumed.

Description

The `RegDelete` command deletes a specified registry entry. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
RegDelete (HKEY_LOCAL_MACHINE  
          , "Software\Hitachi\Script", "CurrentVersion")
```

JP1/Script version

Supported from JP1/Script 01-00.

9.2.4 RegDeleteKey (delete a registry subkey)

Purpose

Deletes a registry subkey.

Syntax

```
RegDeleteKey (RegKey, SubKey, DelKey)
```

Arguments

RegKey

Specify the registry key as one of the following:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

If you specify a registry key other than HKEY_CURRENT_USER, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

DelKey

Specify the key to be deleted as a character string or as a variable that stores this value.

Description

The `RegDeleteKey` command deletes a key in a specified registry subkey, and all the subkeys and entries included in that key. The command returns `True` on successful execution, or `False` if an error occurs.

Example

```
RegDeleteKey(HKEY_LOCAL_MACHINE
             , "Software\Hitachi\Script", "Debug")
```

JP1/Script version

Supported from JP1/Script 01-00.

9.3 Commands for displaying graphics

9.3.1 BitmapShow (draw a bitmap)

Purpose

Draws a bitmap.

Syntax

```
BitmapShow(BmpFileName, [xPos] [, yPos])
```

Arguments

BmpFileName

Specify the full path of the bitmap file as a character string or as a variable that stores this value.

xPos

Specify the horizontal distance from the left side of the screen to the left edge of the window in which the bitmap will be displayed. Specify the distance in pixels, where the left side of the screen is 0, or write a variable that stores this value. If you omit this argument, the window is centered horizontally on the screen. If you specify a negative value, zero is assumed.

yPos

Specify the vertical distance from the top of the screen to the top of the window in which the bitmap will be displayed. Specify the distance in pixels, where the top of the screen is 0, or write a variable that stores this value. If you omit this argument, the window is centered vertically on the screen. If you specify a negative value, zero is assumed.

Description

The `BitmapShow` command draws a specified bitmap. The command returns `True` on successful execution, or `False` if an error occurs.

To erase the bitmap, use the `BitmapHide` command.

Notes

- Do not use this command in a script started as a service; otherwise, an execution error will occur.
- Take care when specifying a bitmap file in the folder set in the `ProgramFiles` environment variable (normally the Program Files folder on the system drive) or `WinDir` (normally the Windows folder on the system drive). For details, see [1.8.2 Command behavior](#).

Example

```
bmpFile = _WIN_ + "ABC.BMP"  
BitmapShow(bmpFile, 100, 100)  
Sleep(10000)  
BitmapHide(bmpFile)
```

JP1/Script version

Supported from JP1/Script 05-00.

9.3.2 BitmapHide (erase a bitmap)

Purpose

Erases a bitmap.

Syntax

```
BitmapHide (BmpFileName)
```

Argument

BmpFileName

Specify the full path of the bitmap file as a character string or as a variable that stores this value.

Description

The `BitmapHide` command erases a displayed bitmap. The command returns `True` on successful execution, or `False` if an error occurs.

`True` is always returned if the specified bitmap is not displayed.

Note

Do not use this command in a script started as a service; otherwise, an execution error will occur.

Example

```
bmpFile = _WIN_+"ABC.BMP"  
BitmapShow(bmpFile, 100, 100)  
Sleep(10000)  
BitmapHide(bmpFile)
```

JP1/Script version

Supported from JP1/Script 05-00.

9.4 Commands for performing evaluations

9.4.1 IsEmptyReg (check whether a registry subkey is empty)

Purpose

Checks whether a registry subkey is empty and returns `True` or `False`.

You can specify registry subkeys to process as exceptions.

Syntax

```
IsEmptyReg (RegKey, SubKey [, ExceptSubKey1, ExceptSubKey2, ...])
```

Arguments

RegKey

Specify the registry key as one of the following:

- `HKEY_CLASSES_ROOT`
- `HKEY_CURRENT_USER`
- `HKEY_LOCAL_MACHINE`
- `HKEY_USERS`

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

ExceptSubKey1 to *ExceptSubKey27*

Specify the registry subkeys to treat as exceptions when the check is performed. Write each exception subkey as a character string or as a variable that stores this value.

Description

The `IsEmptyReg` command checks whether the specified registry subkey contains any other registry subkeys or entries. The command returns `True` if the specified registry subkey is empty, or `False` if not.

If you specify any registry subkeys to treat as exceptions, the command checks for subkeys and entries other than the exception subkey(s).

Example

```
' Check whether registry subkey JP1/Script is empty.
Dim regKey, subKey, isEmp
regKey = HKEY_LOCAL_MACHINE
subKey = "Software\Hitachi\JP1/Script"
isEmp = IsEmptyReg(regKey, subKey, "PathName")
If isEmp = True Then
    MsgBox ("Empty!")
Else
    MsgBox ("Not Empty!")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.4.2 IsExistRegKey (check whether a registry subkey exists)

Purpose

Checks whether a registry subkey exists and returns `True` or `False`.

Syntax

```
IsExistRegKey(RegKey, SubKey)
```

Arguments

RegKey

Specify the registry key as one of the following:

- `HKEY_CLASSES_ROOT`
- `HKEY_CURRENT_USER`
- `HKEY_LOCAL_MACHINE`
- `HKEY_USERS`

SubKey

Specify the registry subkey as a character string or as a variable that stores this value.

Description

The `IsExistRegKey` command checks whether a registry subkey exists. The command returns `True` if the subkey exists, or `False` if not.

Example

```
' If a JP1/Script registry subkey exists,  
' check whether it is empty.  
Dim regKey, subKey  
regKey = HKEY_LOCAL_MACHINE  
subKey = "Software\Hitachi\JP1/Script"  
If IsExistRegKey(regKey, subKey) = True Then  
    isEmp = IsEmptyReg( regKey ,subKey)  
    If isEmp = True Then  
        MsgBox( "Empty !" )  
    Else  
        MsgBox ( "Not Empty !" )  
    End  
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.4.3 IsExistService (check whether a service exists)

Purpose

Checks whether a service exists and returns `True` or `False`.

Syntax

```
IsExistService(ServiceInfoName)
```


Argument

ServiceInfoName

Specify the service information name or service name to check. Write a character string or a variable that stores this value.

Description

The `IsExistService` command checks whether a service has been registered. The command returns `True` if the service has been registered, or `False` if not.

Service information refers to an object that encapsulates information about the service. Although you can specify a service name instead, the service information name is better.

Example

```
' Delete the JP1/Script service if it exists.
ServiceSetValue("JSService", Name::"JP1_Script")
If IsExistService("JSService") Then
    ServiceDelete("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.4.4 IsEmptyGroup (check for shortcuts)

Purpose

Checks whether shortcuts exist in a program group and returns `True` or `False`.

Syntax

```
IsEmptyGroup(GroupName [, RootType])
```

Arguments

GroupName

Specify the program group name as a character string or as a variable that stores this value.

RootType

Specify the program group type with one of the values shown in the following table:

Value	Meaning
<code>Lcl_Program</code>	Program (common program group)
<code>Cur_Program</code>	Program (user-specific program group)

This value is optional. If you omit this value, `Cur_Program` is assumed.

Description

The `IsEmptyGroup` command checks whether a specified program group contains any shortcuts. The command returns `True` if a shortcut exists, or `False` if not.

`False` is always returned if there is no such program group.

Example

```
' Delete the program group "Alphabet" on Windows 7
' if it contains no shortcuts.
```

```
If _OS_ = "WIN_NT6.1" Then
  If IsEmptyGroup("Alphabet") Then
    DeleteGroup("Alphabet")
  End If
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.5 Commands for service operations

9.5.1 ServiceSetValue (set service information)

Purpose

Sets the information required to operate a service as a service information object.

Syntax

```
ServiceSetValue(ServiceInfoName, Keyword1::Value1, Keyword2::Value2, ...)
```

Arguments

ServiceInfoName

Specify the name of the service information object as a character string or as a variable that stores this value. A maximum of 31 single-byte characters can be specified.

Keyword1 to *Keyword11*

Specify each keyword for which a value is to be set.

Value1 to *Value11*

Set a value to each keyword.

Write two colons (: :) between the keyword and corresponding value. You can specify the following keywords and values:

Keyword	Value and meaning
Name	Service name Specify up to 255 characters. Forward-slash (/) and back-slash (\) are not permitted as service name characters.
DispName	Display name Specify up to 255 characters.
Type	Service type SERVICE_WIN32_OWN_PROCESS The service runs as an independent Win32 process. SERVICE_WIN32_SHARE_PROCESS The service shares a Win32 process with another service. SERVICE_KERNEL_DRIVER The service is a Windows device driver. SERVICE_FILE_SYSTEM_DRIVER The service is a Windows file system driver.
Start	Time at which the service starts SERVICE_BOOT_START The operating system loader starts the device driver. This value is valid only when either of the following service types is specified in Type: <ul style="list-style-type: none">SERVICE_KERNEL_DRIVERSERVICE_FILE_SYSTEM_DRIVER SERVICE_SYSTEM_START The IoInitSystem function starts the device driver. This value is valid only when either of the following service types is specified in Type: <ul style="list-style-type: none">SERVICE_KERNEL_DRIVERSERVICE_FILE_SYSTEM_DRIVER SERVICE_AUTO_START

Keyword	Value and meaning
Start	<p>The service control manager automatically starts the device driver or Win32 service at system startup.</p> <p>SERVICE_DEMAND_START</p> <p>The service control manager starts the device driver or Win32 service when the StartService function is invoked from the process.</p> <p>SERVICE_DISABLED</p> <p>The device driver or Win32 service can no longer be started.</p>
Errctl	<p>Error severity if the service fails to start at system startup</p> <p>SERVICE_ERROR_IGNORE</p> <p>The boot program logs the error and continues startup processing.</p> <p>SERVICE_ERROR_NORMAL</p> <p>The boot program logs the error, displays a pop-up message, and continues startup processing.</p> <p>SERVICE_ERROR_SEVERE</p> <p>The boot program logs the error.</p> <p>SERVICE_ERROR_CRITICAL</p> <p>The boot program logs the error if it is possible.</p>
Path	Full path of the service program
Group	<p>Load ordering group to which the service belongs</p> <p>Omit this value if the service does not belong to a group.</p>
DependG	<p>Dependency on a group</p> <p>Specify the load ordering group that must be started before this service.</p>
DependM	<p>Dependency on a service</p> <p>Specify the service that must be started before this service. Omit this value if the service has no dependencies.</p>
StartName	<p>Service account name</p> <p>SERVICE_WIN32_OWN_PROCESS specified in Type</p> <p>The service is logged on as an account name in the form <i>DomainName\Username</i>. If the service account belongs to the built-in domain, specify the name as <i>\UserName</i>.</p> <p>SERVICE_WIN32_SHARE_PROCESS specified in Type</p> <p>The LocalSystem account must be used. If you omit this value, the service is logged on under the LocalSystem account.</p> <p>SERVICE_KERNEL_DRIVER, or SERVICE_FILE_SYSTEM_DRIVER, specified in Type</p> <p>This value is the name of the driver object (<i>\FileSystem\Rdr</i>, or <i>\Driver\Xns</i>, for example) used when the I/O system loads the device driver. If you omit this value, the I/O system creates a default object name based on the service name, and starts the driver using this name.</p>
Password	<p>Account password</p> <p>SERVICE_WIN32_OWN_PROCESS, or SERVICE_WIN32_SHARE_PROCESS, specified in Type</p> <p>Specify the password of the service account set in StartName.</p> <p>SERVICE_KERNEL_DRIVER, or SERVICE_FILE_SYSTEM_DRIVER, specified in Type</p> <p>This value is ignored.</p> <p>Omit this value if the service has no password.</p>

Description

The execution result of the `ServiceSetValue` command is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Service information refers to an object that encapsulates information about the service. You can set and acquire these attributes using the above keywords.

Before you use any other service operation command, set the keywords required in the service information, using the `ServiceSetValue` command. You can omit unnecessary keywords.

Example

```
OutDir = "C:\Program Files\Hitachi\Script\BIN\"
ServiceSetValue("JSService", Name::"JP1_Script"
, DispName::"JP1/Script"
, Type::SERVICE_WIN32_OWN_PROCESS
, Start::SERVICE_AUTO_START
, Errctl::SERVICE_ERROR_NORMAL
, Path::OutDir+"SPTHSV.EXE")
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.2 ServiceGetValue (get values in the service information)

Purpose

Gets values stored in a service information object.

Syntax

```
ServiceGetValue(ServiceInfoName, Keyword1::ValueBuff1,
Keyword2::ValueBuff2, ...)
```

Arguments

ServiceInfoName

Specify the name of the service information object as a character string or as a variable that stores this value.

Keyword1 to *Keyword11*

Specify each keyword.

ValueBuff1 to *ValueBuff11*

Specify a variable for storing the value of each keyword.

The keywords you can specify in *Keyword1* to *Keyword11*, and the values set in *ValueBuff1* to *ValueBuff11* are as follows:

Keyword	Value and meaning
Name	Service name
DispName	Display name
Type	Service type SERVICE_WIN32_OWN_PROCESS The service runs as an independent Win32 process. SERVICE_WIN32_SHARE_PROCESS The service shares a Win32 process with another service. SERVICE_KERNEL_DRIVER The service is a Windows NT device driver. SERVICE_FILE_SYSTEM_DRIVER The service is a Windows NT file system driver.
Start	Time at which the service starts SERVICE_BOOT_START The operating system loader starts the device driver. SERVICE_SYSTEM_START

Keyword	Value and meaning
Start	<p>The IoInitSystem function starts the device driver.</p> <p>SERVICE_AUTO_START</p> <p>The service control manager automatically starts the device driver or Win32 service at system startup.</p> <p>SERVICE_DEMAND_START</p> <p>The service control manager starts the device driver or Win32 service when the StartService function is invoked from the process.</p> <p>SERVICE_DISABLED</p> <p>The device driver or Win32 service can no longer be started.</p>
Errctl	<p>Error severity if the service fails to start at system startup</p> <p>SERVICE_ERROR_IGNORE</p> <p>The boot program logs the error and continues startup processing.</p> <p>SERVICE_ERROR_NORMAL</p> <p>The boot program logs the error, displays a pop-up message, and continues startup processing.</p> <p>SERVICE_ERROR_SEVERE</p> <p>The boot program logs the error.</p> <p>SERVICE_ERROR_CRITICAL</p> <p>The boot program logs the error if it is possible.</p>
Path	Full path of the service program
Group	Load ordering group to which the service belongs
DependG	Dependency on a group
DependM	Dependency on a service
StartName	Service account name
Password	Account password

Description

The ServiceGetValue command acquires the stored service information for the required keywords only. The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Example

```
' Get information about the JP1/Script service.
Dim svDispName, svType, svStart, svErrctl, svPath
' Set JP1/Script "JSService" service information.
ServiceSetValue("JSService", Name::"JP1_Script")
' Get the settings for "JSService" service information.
ServiceQuery("JSService")
' Get the settings stored in "JSService" service information,
' and store them in variables.
ServiceGetValue("JSService", DispName::svDispName _
                , Type::svType _
                , Start::svStart _
                , Errctl::svErrctl _
                , Path::svPath)
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.3 ServiceCreate (register a service)

Purpose

Registers a service.

Syntax

```
ServiceCreate(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be registered. Write a character string or a variable that stores this value.

Description

The `ServiceCreate` command registers a service in the system, based on the information stored in a service information object.

As a minimum requirement for registering a service, you must set values in the `Name`, `DispName`, `Type`, `Start`, `Errctl`, and `Path` keywords. For details, see [9.5.1 ServiceSetValue \(set service information\)](#).

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

To register a service, you must log on with an account that has system administrator privileges.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Register the JP1/Script service if it does not already exist.
ServiceSetValue("JSService", Name::"JP1_Script")
If IsExistService("JSService") = False Then
  OutDir = "C:\Program Files\Hitachi\Script\BIN\"
  ServiceSetValue("JSService", Name::"JP1_Script"
    , DispName::"JP1/Script"
    , Type::SERVICE_WIN32_OWN_PROCESS
    , Start::SERVICE_AUTO_START
    , Errctl::SERVICE_ERROR_NORMAL
    , Path::OutDir+"SPTHSV.EXE")
  ServiceCreate("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.4 ServiceDelete (delete a service)

Purpose

Deletes a service.

Syntax

```
ServiceDelete(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be deleted. Write a character string or a variable that stores this value.

Description

The `ServiceDelete` command deletes a service registered in the system, using the value stored in the keyword `Name` in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Important note

Do not execute this command for services that cannot be stopped. This command issues a stop request to the specified service to be deleted, and will not terminate until the specified service stops so that it can be deleted.

Example

```
' Delete the JP1/Script service if it exists.
ServiceSetValue("JSService", Name::"JP1_Script")
If IsExistService("JSService") Then
    ServiceDelete("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.5 ServiceStart (start a service)

Purpose

Starts a service.

Syntax

```
ServiceStart(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be started. Write a character string or a variable that stores this value.

Description

The `ServiceStart` command starts a service registered in the system, using the value stored in the keyword `Name` in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `LOGON_FAILED` if the service fails to start due to an invalid account. `False` is returned for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Important note

Do not execute this command for services that cannot start. This command issues a start request to the specified service, and will not terminate until the specified service has started.

Example

```
' Register and start the JP1/Script service if it does
' not already exist.
ServiceSetValue("JSService", Name::"JP1_Script")
If IsExistService("JSService") = False Then
  OutDir = "C:\Program Files\Hitachi\Script\BIN\"
  ServiceSetValue("JSService", Name::"JP1_Script"
    , DispName::"JP1/Script"
    , Type::SERVICE_WIN32_OWN_PROCESS
    , Start::SERVICE_AUTO_START
    , Errctl::SERVICE_ERROR_NORMAL
    , Path::OutDir+"SPTHSV.EXE")
  ServiceCreate("JSService")
End
ServiceStart("JSService")
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.6 ServiceStop (stop a service)

Purpose

Stops a service.

Syntax

```
ServiceStop(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be stopped. Write a character string or a variable that stores this value.

Description

The `ServiceStop` command stops an active service, using the value stored in the keyword `Name` in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Important note

Do not execute this command for services that cannot be stopped. This command issues a stop request to the specified service, and will not terminate until the specified service stops.

Example

```
' Stop the active JP1/Script service.
Dim svStatus
ServiceSetValue("JSService", Name::"JP1_Script")
ServiceRefer("JSService", svStatus)
If svStatus = SERVICE_RUNNING Then
    ServiceStop("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.7 ServicePause (halt a service)

Purpose

Temporarily halts an active service.

Syntax

```
ServicePause(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be halted. Write a character string or a variable that stores this value.

Description

The `ServicePause` command temporarily halts an active service, using the value stored in the keyword `Name` in the service information.

This command internally executes the `ServiceControl` command.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Halt the active JP1/Script service.
Dim svStatus
ServiceSetValue("JSService", Name::"JP1_Script")
ServiceRefer("JSService", svStatus)
If svStatus = SERVICE_RUNNING Then
    ServicePause("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.8 ServiceContinue (resume a halted service)

Purpose

Resumes a halted service.

Syntax

```
ServiceContinue (ServiceInfoName)
```

Argument

ServiceInfoName

Specify the service information name of the service to be resumed. Write a character string or a variable that stores this value.

Description

The `ServiceContinue` command resumes a halted service, using the value stored in the keyword `Name` in the service information.

This command internally executes the `ServiceControl` command.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Resume the halted JP1/Script service.
Dim svStatus
ServiceSetValue("JSService", Name::"JP1_Script")
ServiceRefer("JSService", svStatus)
If svStatus = SERVICE_PAUSED Then
    ServiceContinue("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.9 ServiceChange (change a setting for a service)

Purpose

Changes the information set for a service.

Syntax

```
ServiceChange (ServiceInfoName)
```

Argument

ServiceInfoName

Specify the name of the service information in which the changes are stored. Write a character string or a variable that stores this value.

Description

The `ServiceChange` command changes a setting in the service information for the current service.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Change the setting (startup type) for the JP1/Script
' service.
Dim svStart
' Set JP1/Script for "JSService" service information.
ServiceSetValue("JSService", Name::"JP1_Script")
' Get the settings for "JSService" service information.
ServiceQuery("JSService")
' Get the setting (startup type) stored in "JSService" service
' information, and store it in the variable.
ServiceGetValue("JSService", Start::svStart)
' If the startup type of the service is automatic startup,
' Change it to manual startup.
If svStart = 3 Or svStart = SERVICE_AUTO_START Then
    svStart = SERVICE_DEMAND_START
    ServiceSetValue("JSService", Start::svStart)
    ServiceChange("JSService")
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.10 ServiceQuery (get the information set for an active service)

Purpose

Gets the information set for an active service.

Syntax

```
ServiceQuery(ServiceInfoName)
```

Argument

ServiceInfoName

Specify the name of the service information to be acquired. Write a character string or a variable that stores this value.

Description

The `ServiceQuery` command gets the settings for an active service and stores them in the service information.

To acquire service information, the service name must be set in the `Name` keyword in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Example

```
' Get the setting (service type) for the JP1/Script service.
Dim svType
' Set JP1/Script for "JSService" service information.
ServiceSetValue("JSService", Name::"JP1_Script")
' Get the settings for "JSService" service information.
ServiceQuery("JSService")
' Get the setting (service type) stored in "JSService" service
' information, and store it in the variable.
ServiceGetValue("JSService", Type::svType)
MessageBox("JP1/Script service type: "+svType)
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.11 ServiceRefer (get the current status of a service)

Purpose

Gets the current status of a service.

Syntax

```
ServiceRefer(ServiceInfoName, StatusBuff)
```

Arguments

ServiceInfoName

Specify the name of the service information object as a character string or as a variable that stores this value.

StatusBuff

Specify a variable for storing the current status. This variable stores one of the following values:

Value	Meaning
SERVICE_STOPPED	The service is not running.
SERVICE_START_PENDING	The service is starting.
SERVICE_STOP_PENDING	The service is stopping.
SERVICE_RUNNING	The service is running.
SERVICE_CONTINUE_PENDING	The service has not yet resumed.
SERVICE_PAUSE_PENDING	The service has not yet halted.
SERVICE_PAUSED	The service has temporarily halted.

Description

The `ServiceRefer` command gets the current status of a service and stores the result in a variable.

To acquire the service status, the service name must be set in the `Name` keyword in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Example

```
' Get the current status of the JP1/Script service,  
' and activate the service.  
Dim svStatus  
ServiceSetValue("JSService", Name::"JP1_Script")  
ServiceRefer("JSService", svStatus)  
If svStatus = SERVICE_STOPPED Then  
    ServiceStart("JSService")  
End
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.12 ServiceControl (send a control command to a service)

Purpose

Sends a control command to a service.

Syntax

```
ServiceControl(ServiceInfoName, Control)
```

Arguments

ServiceInfoName

Specify the name of the service information object as a character string or as a variable that stores this value.

Control

Specify the control to send to the service as one of the following values:

Value	Meaning
SERVICE_CONTROL_STOP	Stop the service. Unlike the <code>ServiceStop</code> command, this control command results in an error if the service is inactive.
SERVICE_CONTROL_PAUSE	Halt the service. Same function as the <code>ServicePause</code> command.
SERVICE_CONTROL_CONTINUE	Resume the service. Same function as the <code>ServiceContinue</code> command.

Description

The `ServiceControl` command sends a control command to a service, using the value stored in the keyword `Name` in the service information.

The service name of the service to be controlled must be set in the `Name` keyword in the service information.

The execution result is stored in the `_SVC_RTN_` reserved variable. The command returns `True` on successful execution, or `False` for all other states.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Halt the JP1/Script service.
ServiceSetValue("JSService", Name::"JP1_Script")
ServiceControl("JSService", SERVICE_CONTROL_PAUSE)
```

JP1/Script version

Supported from JP1/Script 01-00.

9.5.13 GetServiceName (get the service name from a service display name)

Purpose

Gets the service name from the display name of a service.

Syntax

```
GetServiceName(DispName)
```

Arguments

DispName

Specify the display name of the service as a character string or as a variable that stores this value.

Description

The `GetServiceName` command gets the service name from the display name of a specified service. The command returns the service name on successful execution. A zero-length string ("") is returned if an error occurs.

Example

```
' Halt the service that has the display name "JP1/Script".
svName = GetServiceName("JP1/Script")
If IsEmpty(svName) = False Then
    ServiceSetValue("JSService", Name::svName)
    If IsExistService("JSService") Then
        ServiceStop("JSService")
    End
End
```

JP1/Script version

Supported from JP1/Script 05-10.

9.6 Command for calling an external program

9.6.1 CallDll (call a DLL file)

Purpose

Calls a dynamic-link library (DLL) file.

Syntax

```
CallDll(DllFileName, FunctionName, Param1, Param2, ...)
```

Arguments

DllFileName

Specify the DLL file name as a character string or as a variable that stores this value.

FunctionName

Specify the external function to invoke. Write a character string or a variable that stores this value.

Param1 to Param32

Specify each of the function parameters as a character string or as a variable that stores this value.

In JP1/Script 06-00 and later versions, you can specify an array variable that stores all the required parameters.

Description

The `CallDll` command loads the specified DLL and calls an external function. The command returns `True` on successful execution, or `False` if an error occurs.

Use the following interface for the external function. The external references must be declared in the `EXPORTS` section of the module definition file (`.DEF`).

Declaration

```
BOOL WINAPI MyFunc(HWND hParent, int argc, char * argv[], int *  
rtnc, char * * rtnv[]);
```

Arguments

- `HWND hParent;`
Handle of the parent window
- `int argc;`
Number of parameters to be passed from the script
- `char * argv[];`
Array that stores the parameters to be passed from the script
- `int * rtnc;`
Pointer to the number of strings to return to the script.
- `char * * rtnv[];`
Array that stores the number of strings to return to the script.

Return values

The return value of the `MyFunc` external function is stored in the `_DLL_RTN_` reserved variable. When `True` is set in `_DLL_RTN_`, you can use the `_RTNxx_` reserved variable in your script to reference the array holding

the strings to be returned to the script, stored in the `rtnv` argument. Here, `xx` is a sequential number starting from 00, up to the number stored in the `rtnc` argument.

When `False` is set in `_DLL_RTN_`, the `_RTNxx_` reserved variable is undefined.

To enable the returned string and string array to be referenced after the function completes, do not store these values in a local buffer within the scope of the function.

Note

The loaded DLL file will be unloaded when the command terminates. If you do not want to unload the DLL file, set 1 in the following registry key. The loaded DLL file will not be unloaded until execution of the script terminates.

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\HITACHI\JP1/Script\SPTX

Value name

CallDllUnloadMode

Value datatype

REG_DWORD

Value

0: Unload DLL files.

1: Do not unload DLL files.

When the setting takes effect

The setting takes effect the next time the script file is executed.

Important note

Note the following when you specify not to unload DLL files:

- The `DllMain` function, which is a DLL entry point function, is called from the Windows system and executed when a DLL file is loaded. Therefore, if you execute the `CallDll` command multiple times with the same DLL file specified, the `DllMain` function is executed during the first execution of the command, which loads the DLL file. The `DllMain` function is not executed at the second or subsequent executions of the command because no DLL file is loaded.
- The external variables used in the DLL are initialized when the DLL file is loaded. Therefore, if you execute the `CallDll` command multiple times with the same DLL file specified, the external variables are initialized during the first execution of the command, which loads the DLL file. The external variables are not initialized at the second or subsequent executions of the command because the DLL file is not loaded.
- To specify DLL files with the same name and different entities, use absolute paths. If you use relative paths to specify such DLL files, the DLL files will not be loaded even if you specify the path to the executable folder in the `SetPath` command. This is because the Windows system identifies these DLL files as the same file.
- As far as possible, use a DLL file that contains a group of external functions that will be called by the `CallDll` command. If you use the `CallDll` command with different DLL files specified, many DLL files will be loaded during execution of the script execution process. As a result, memory might be insufficient.

Example

```
' Function part of TEST.DLL(Favorite.c)
#include "windows.h"
```

```

#define FAV_SPORT1 "SKI"
#define FAV_SPORT2 "BASKETBALL"
#define FAV_FOOD1 "STEAK"
#define FAV_FOOD2 "PASTA"
char * g_ret[2];

BOOL WINAPI GetFavorite(HWND hParent, int argc, char * argv[], int *
rtnc, char ** rtnv[])
{
    if(lstrcmp(argv[0], "SPORT") == 0)
    {
        g_ret[0] = FAV_SPORT1;
        g_ret[1] = FAV_SPORT2;
    }
    else if(lstrcmp(argv[0], "FOOD") == 0)
    {
        g_ret[0] = FAV_FOOD1 ;
        g_ret[1] = FAV_FOOD2;
    }

    *rtnc = 2;
    *rtnv = g_ret;

    return(TRUE);
}

' Processing(abc.SPT) at the script side
Dim Quest
Quest = "SPORT"
CallDll(_BIN_"TEST.DLL", "GetFavorite", Quest)
Dim Msg
Msg = "My favorite" +Quest+" are "+_RTN00_" and "+_RTN01_+"."
MessageBox(Msg, OK, , Information)
Exit(0)

```

JP1/Script version

Supported from JP1/Script 01-00.

9.7 Commands for performing shortcuts

9.7.1 MakeGroup (create a program group)

Purpose

Creates a program group.

Syntax

```
MakeGroup (GroupName [, RootType])
```

Arguments

GroupName

Specify the program group name as a character string or as a variable that stores this value.

If you specify `Lcl_Program`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

RootType

Specify the program group type with one of the values shown in the following table:

Value	Meaning
<code>Lcl_Program</code>	Program (common program group)
<code>Cur_Program</code>	Program (user-specific program group)

This value is optional. If you omit this value, `Cur_Program` is assumed.

Description

The `MakeGroup` command creates a specified program group. The command returns `True` on successful execution, or `False` if an error occurs.

If the specified program group already exists, the command returns `True` without doing anything.

Example

```
' Create the program group "Alphabet" on Windows 7
' and enter the shortcut "ABC" in that group.
If _OS_ = "WIN_NT6.1" Then
    MakeGroup("Alphabet")
    MakeShortcut(Cur_Program, "Alphabet\ABC", _
        _BIN_+"ABC.SPT")
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.7.2 DeleteGroup (delete a program group)

Purpose

Deletes a program group.

Syntax

```
DeleteGroup(GroupName [, RootType])
```

Arguments

GroupName

Specify the program group name as a character string or as a variable that stores this value.

If you specify `Lcl_Program`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

RootType

Specify the program group type with one of the values shown in the following table:

Value	Meaning
<code>Lcl_Program</code>	Program (common program group)
<code>Cur_Program</code>	Program (user-specific program group)

This value is optional. If you omit this value, `Cur_Program` is assumed.

Description

The `DeleteGroup` command deletes a specified program group. The command returns `True` on successful execution, or `False` if an error occurs.

If there is no such program group, the command returns `True` without doing anything.

Example

```
' Delete the program group "Alphabet" on Windows 7
If _OS_ = "WIN_NT6.1" Then
    DeleteGroup("Alphabet")
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.7.3 MakeShortcut (create a shortcut)

Purpose

Creates a shortcut.

Syntax

```
MakeShortcut(RootPath, SubPath, LinkPath, [Param], [WorkDirPath],  
[IconPath], [IconIndex] [, IconFlag])
```

Arguments

RootPath

Specify the location where the shortcut is to be created.

This value is one of the following.

Value	Meaning
<code>None</code>	Create in the current executable folder.

Value	Meaning
Lcl_Desktop	Desktop (common group)
Lcl_Startmenu	Start menu (common group)#1
Lcl_Program	Program (common group)#2
Lcl_Startup	Startup (common group)#3
Cur_Desktop	Desktop (user-specific group)
Cur_Startmenu	Start menu (user-specific group)#1
Cur_Program	Program (user-specific group)#2
Cur_Startup	Startup (user-specific group)#3

#1 Windows **Start** menu

#2 Windows **Start**, and then **Programs**

#3 Windows **Start**, **Programs**, and then **Startup**

If you specify `Lcl_Desktop`, `Lcl_Startmenu`, `Lcl_Program`, or `Lcl_Startup`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

SubPath

Specify the shortcut name as a character string or as a variable that stores this value.

If you set `Lcl_Program` or `Cur_Program` as the shortcut location in *RootPath*, specify the shortcut name in the following form:

program-group\shortcut-name

LinkPath

Specify the path to the executable file that the shortcut points to. Write a character string or a variable that stores this value.

Param

Specify the necessary parameters for executing the file specified in *LinkPath*. Write each parameter as a string, or specify a variable that stores this value.

WorkDirPath

Specify the work folder name as a character string or as a variable that stores this value.

This value is optional. If you omit this value, the folder of the path specified in *LinkPath* is assumed.

IconPath

Specify the name of the icon file containing the icons used for shortcuts. Write the path as a character string or as a variable that stores this value.

This value is optional. If you omit this value, the path specified in *LinkPath* is assumed.

IconIndex

Specify the index number of the icon inside the icon file specified in *IconPath*. Specify 0 or a number, or a variable that stores this value.

This value is optional. If you omit this value, 0 is assumed.

IconFlag

Specify how to display the window when the shortcut is activated. Set one of the following values:

Value	Meaning
Normal	Normal window

Value	Meaning
Min	Minimized into a button in the task bar
Max	Maximized to full-screen size

This value is optional. If you omit this value, Normal is assumed.

Description

The `MakeShortcut` command creates a specified shortcut. The command returns `True` on successful execution, or `False` if an error occurs.

If you specify a shortcut that already exists, the command deletes and recreates the shortcut with the specified parameters.

Example

```
' Create shortcut "ABC.SPT" to be launched at
' Windows 7 startup
If _OS_ = "WIN_NT6.1" Then
    MakeShortcut(Cur_Startup, "ABC", _BIN_+"ABC.SPT" _
                , , , _TEMP_ , , , Min)
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.7.4 DeleteShortcut (delete a shortcut)

Purpose

Deletes a shortcut.

Syntax

```
DeleteShortcut(RootPath, SubPath)
```

Arguments

RootPath

Specify the location where the shortcut is located.

This value is one of the following.

Value	Meaning
None	In the current executable folder
Lcl_Desktop	Desktop (common group)
Lcl_Startmenu	Start menu (common group) ^{#1}
Lcl_Program	Program (common group) ^{#2}
Lcl_Startup	Startup (common group) ^{#3}
Cur_Desktop	Desktop (user-specific group)
Cur_Startmenu	Start menu (user-specific group) ^{#1}
Cur_Program	Program (user-specific group) ^{#2}

Value	Meaning
Cur_Startup	Startup (user-specific group) ^{#3}

#1 Windows **Start** menu

#2 Windows **Start**, and then **Programs**

#3 Windows **Start**, **Programs**, and then **Startup**

If you specify `Lcl_Desktop`, `Lcl_Startmenu`, `Lcl_Program`, or `Lcl_Startup`, execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

SubPath

Specify the shortcut name as a character string or as a variable that stores this value.

If you set `Lcl_Program` or `Cur_Program` as the shortcut location in *RootPath*, specify the shortcut name in the following form:

program-group\shortcut-name

Description

The `DeleteShortcut` command deletes a specified shortcut. The command returns `True` on successful execution, or `False` if an error occurs.

If there is no such shortcut, the command returns `True` without doing anything.

Example

```
' Delete shortcut "ABC" from Windows 7 startup.
If _OS_ = "WIN_NT6.1" Then
  DeleteShortcut(Cur_Startup, "ABC")
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.8 Commands for process monitoring

9.8.1 GetProcessCount (get the number of process activations)

Purpose

Gets the number of activations of a process.

Syntax

```
GetProcessCount (ProcessName [, ProcessIdBuff])
```

Arguments

ProcessName

Specify the process name as a character string or as a variable that stores this value.

Specify the process name without a folder name or an extension such as .EXE.

ProcessIdBuff

Specify a variable for storing the process ID(s).

In JP1/Script 06-00 and later versions, you can specify a dynamic one-dimensional array variable.

Omit this argument if not required.

When the process specified in *ProcessName* is activated concurrently from two or more calling processes, the process IDs are sorted in start order and stored in either of the following ways:

- If you specify a variable in *ProcessIdBuff*, the process IDs are stored as a string delimited with semi-colons (;).
- If you specify an array variable in *ProcessIdBuff*, the process IDs are stored in order from the first element. Afterwards, the size of the array variable is set to the number of process IDs.

Description

The `GetProcessCount` command gets the number of activations of a specified process. On successful execution, the command returns the process count. If you specify the *ProcessIdBuff* argument, the command also returns the process ID(s).

If the specified process has not been activated, the command returns zero and stores a zero-length string ("") in *ProcessIdBuff*. If an error occurs, a zero-length string ("") is returned.

Supplement

- Due to a restriction in the Windows specifications, a complete name cannot be acquired for processes activated with a local system account. For this reason, the following problem occurs if the `GetProcessCount` command is executed from an account other than the system account by specifying the name of a process activated with the local system account: If the specified process name contains more than 15 characters (including the extension), the number of activations and the process ID(s) are not returned, even if the process is active. To avoid this situation, specify no more than the first 15 characters of the process name, including the extension, in *ProcessName*. For details, see Example 2.
- Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Important note

Execute the script file as a user who has administrator permissions.

Example 1

```
' List the process IDs for process "ABC".
Dim pCnt, pIdBuff, seppId
pCnt = GetProcessCount("ABC", pIdBuff)
For i = 1 To pCnt
    seppId = SeparateStr(pIdBuff, ";", i)
    Message(Target_File, _SCF_"PIDLIST.TXT", seppId)
Next
```

Example 2

```
Dim PLName, Pname, PnameWork, Rc, IdBuff
PLName = "ABCDEFGHijkl"
Pname = "ABCDEFGHijkl.exe"
' First execute with normal specification method
' (specify process name).
Rc = GetProcessCount(PLName, IdBuff)
' If number of activations is 0, execute
' processing that considers process is activated
' from local system account.
If Rc = 0 Then
    If Len(Pname) > 15 Then
        PnameWork = Left(Pname, 15)
    End If
    Rc = GetProcessCount(PnameWork, IdBuff)
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.8.2 GetProcessInfo (get process information)

Purpose

Gets process information for a specified process ID.

Syntax

```
GetProcessInfo(ProcessId, [StartTimeBuff] [, ProcessorTimeBuff])
```

Arguments

ProcessId

Specify the process ID as a number or as a variable that stores this value.

This process ID is the value returned in the buffer argument of the `GetProcessCount` command.

In JP1/Script 06-00 and later versions, you can specify the `_EXEC_ID_` reserved variable set at execution of the `Exec` or `NetExec` command. In this case, the `GetProcessInfo` command acquires process information for an executable file called by an `Exec` or `NetExec` command specified not to wait for completion of the executable file (*Flag* argument set to `False`).

StartTimeBuff

Specify a variable for storing the process start time. The time is returned in `hh:mm:ss` format. Omit this argument if not required.

ProcessorTimeBuff

Specify a variable for storing the process operation time (seconds). Omit this argument if not required.

Description

The `GetProcessInfo` command gets process information for a specified process ID, and stores the information in the specified variables. The command returns `True` on successful execution, or `False` if an error occurs.

Note

Execute the script file as a user who has administrator permissions. For details, see [1.8.2 Command behavior](#).

Example

```
' Get information about active process "ABC".
Dim pCnt, pIdBuff, seppId, infBuff1, infBuff2
pCnt = GetProcessCount("ABC", pIdBuff)
For i = 1 To pCnt
    seppId = SeparateStr(pIdBuff, ";", i)
    GetProcessInfo(seppId, infBuff1, infBuff2)
    :
    :
Next
```

JP1/Script version

Supported from JP1/Script 05-20.

9.8.3 TerminateProcess (forcibly terminate a process)

Purpose

Forcibly terminates a process specified by process ID.

Syntax

```
TerminateProcess(ProcessId [, Code])
```

Arguments

ProcessId

Specify the process ID as a number or as a variable that stores this value.

This process ID is the value returned in the buffer argument of the `GetProcessCount` command.

In JP1/Script 06-00 and later versions, you can specify the `_EXEC_ID_` reserved variable set at execution of the `Exec` or `NetExec` command. In this case, the `GetProcessInfo` command acquires process information for an executable file called by an `Exec` or `NetExec` command specified not to wait for completion of the executable file (*Flag* argument set to `False`).

Code

Specify the exit code for the process to be terminated. Write a number or a variable that stores this value.

When this value is omitted, 0 is assumed.

Description

The `TerminateProcess` command forcibly terminates a process specified by process ID. The command returns `True` on successful execution, or `False` if an error occurs.

`True` is always returned if there is no such process ID.

Notes

- At execution of this command, a message is output to the application log at the Event Viewer. The event ID is 7. Output to the event log is suppressed when the parameter /NOEVLOG or /NOEVLOG (7) is set in the command line defined in the execution environment file or in the command line set in the registry.
- Execute the script file as a user who has administrator permissions. For details, see *1.8.2 Command behavior*. If you do not execute the script file as a user who has administrator permissions, you cannot terminate processes being executed by other users (but can forcibly terminate script processes).

Example

```
' Forcibly terminate all active "ABC" processes other
' than the "ABC" process that started first.
Dim pCnt, pIdBuff, seppId
pCnt = GetProcessCount("ABC", pIdBuff)
If pCnt > 1 Then
  For i = 2 To pCnt
    seppId = SeparateStr(pIdBuff, ";", i)
    TerminateProcess(seppId)
  Next
End If
```

JP1/Script version

Supported from JP1/Script 05-20.

9.9 Other commands

9.9.1 ExitWindows (terminate a script and log off or shut down Windows)

Purpose

Terminates script execution and logs off from Windows, or turns off or reboots the Windows system.

Syntax

```
ExitWindows([Code] [, Option1, Option2])
```

Arguments

Code

Specify the exit code as a number or as a variable that stores this value.

If you omit this argument, zero is assumed.

Option1

Specify the shutdown type as one of the following values:

Value	Meaning
Logoff	Shut down all active processes, then log off the current user.
Poweroff, Shutdown	Shut down and power off the system. The right to shut down the system is required. The system must support software-driven poweroff. Depending on the Do not turn off system power after a Windows system shutdown has occurred. setting in the Windows group policy settings, the user can power off the system after it is shut down.
Reboot	Restart the system after shutdown. The right to shut down the system is required.
Force	Forcibly terminate the process without displaying an application did not respond message if an application fails to respond to a shutdown.

This value is optional. If you omit this value, `Logoff` is assumed.

Option2

If you specified a value other than `Force` in *Option1*, specify the following optional value as the shutdown operation to be used in conjunction with *Option1*:

Value	Meaning
Force	Forcibly terminate the process without displaying an application did not respond message if an application fails to respond to a shutdown.

Description

The `ExitWindows` command terminates script execution and exits Windows using the specified type of shutdown operation.

For details on JP1/Script exit codes, see [6.1.12 JP1/Script exit codes](#).

Note

At execution of this command, a message is output to the application log at the Event Viewer. The event ID is 5. Output to the event log is suppressed when the parameter `/NOEVLOG` or `/NOEVLOG (5)` is set in the command line defined in the execution environment file or in the command line set in the registry.

Important note

- If you execute the `ExitWindows` command while another program or service is running, that program or service may malfunction. Exercise care when you use the `ExitWindows` command.
- If you execute multiple `ExitWindows` commands at the same time, the operation specified for *Option1* might be disabled or the OS might stop.
- If you execute the `ExitWindows` command by using a script file started from JP1/AJS2 or JP1/AJS3, the JP1/AJS2 or JP1/AJS3 environment might be damaged.
- The `ExitWindows` command does not function in the following cases:
 - The command is executed from a service for which an account without permission to shut down the system has been set.
 - The command is executed from the logon space of a computer to which the user logged on with an account without permission to shut down the system.
 - `Force` is not specified in the command's *Option1* or *Option2* and the running application rejects the shutdown request. However, if the command is executed from a service, the command takes effect regardless of whether `Force` is specified (except `Logoff`).
 - `Force` is not specified in the command's *Option1* or *Option2* and the computer to which the user logged on is locked. However, if the command is executed from a service, the command takes effect regardless of whether `Force` is specified (except `Logoff`).
 - `Logoff` is specified in *Option1* and the command is executed from a service.

Example

```
MessageBox("Do you want to terminate the script and  
          shut down the system?", OKCancel)  
If _MSG_RTN_ = OK Then  
    ExitWindows(0, Shutdown)  
Else  
    Exit(0)  
End
```

JP1/Script version

Supported from JP1/Script 05-10.

9.9.2 SetRetryMode (set the lock error retry function)

Purpose

Sets the lock error retry function.

Syntax

```
SetRetryMode(Count [, WaitTime])
```

Arguments

Count

Specify the retry count as a number in the range from 0 to 100, or as a variable that stores this value.
If you specify 0, the retry function is canceled.

WaitTime

Specify retry wait time as a number (milliseconds) in the range from 100 to 60,000, or as a variable that stores this value.

This value is optional. If you omit this value, 100 is assumed.

Description

The `SetRetryMode` command retries access according to the specified retry count and retry wait time if a lock error occurs in a command or statement that can be used with the lock error retry function.

The specified retry count and retry wait time are effective until the `ResetRetryMode` command is executed.

Example

```
Dim BkupFileName, FileNo
TempFile( BkupFileName, "BUP", _TEMP_ )
Copy( _TEMP + "MODE.INF", BkupFileName )
DeleteFile( _TEMP_ + "MODE.INF" ) ' Disk write might delay
SetRetryMode( 5, 100 ) ' Set retry.
FileNo = TextOpen( _TEMP_ + "MODE.INF", CREATE )
ResetRetryMode ' Cancel retry.
If FileNo = 0 Then
    MsgBox( _TEMP_ + "Failed to open MODE.INF", OK )
    Exit 2
Else
    TextWrite( FileNo, "Mode=ReadOnly" )
End If
TextClose( FileNo )
```

JP1/Script version

Supported from JP1/Script 10-00.

9.9.3 ResetRetryMode (cancel the lock error retry function)

Purpose

Cancels the lock error retry function.

Syntax

```
ResetRetryMode
```

This command has no arguments.

Description

The `ResetRetryMode` command cancels the retry operation to be performed if a lock error occurred in a command or statement that can be used with the lock error retry function.

Example

```
Dim BkupFileName, FileNo
TempFile( BkupFileName, "BUP", _TEMP_ )
Copy( _TEMP + "MODE.INF", BkupFileName )
DeleteFile( _TEMP_ + "MODE.INF" ) ' Disk write might delay
SetRetryMode( 5, 100 ) ' Set retry.
FileNo = TextOpen( _TEMP_ + "MODE.INF", CREATE )
ResetRetryMode ' Cancel retry.
If FileNo = 0 Then
    MsgBox( _TEMP_ + "Failed to open MODE.INF", OK )
```

```
Exit 2
Else
  TextWrite( FileNo, "Mode=ReadOnly" )
End If
TextClose( FileNo )
```

JP1/Script version

Supported from JP1/Script 10-00.

9.9.4 SetTrialOpenMode (set the trial-open function)

Purpose

Sets the trial-open function.

Syntax

```
SetTrialOpenMode ( Count [,WaitTime])
```

Arguments

Count

Specify the trial-open retry count as a number in the range from 0 to 100, or as a variable that stores this value. If you specify 0, the trial-open function is canceled.

WaitTime

Specify the trial-open retry interval as a number (milliseconds) in the range from 100 to 60,000, or as a variable that stores this value.

This value is optional. If you omit this value, 100 is assumed.

Description

The `SetTrialOpenMode` command tries to open a file if a lock error occurs during an attempt to open a file that was closed by the `TextClose` command.

The specified trial-open retry count and trial-open retry interval are effective until the `ResetTrialOpenMode` command is executed.

Example

```
Dim file1, Buff1
file1 = TextOpen( _BIN_ + "Error.log", ReadWrite )
If file1 = 0 Then
  MessageBox( _BIN_ + "Failed to open", OK )
  Exit 2
End If
TextSeek( file1, ToEnd )
TextWrite( file1, "A lock error occurred." )
SetTrialOpenMode( 2, 6000 ) ' Set trial-open.
TextClose( file1 ) ' Disk write might delay
ResetTrialOpenMode ' Cancel trial-open.
Exec( "MessageOutput.EXE", True, _BIN_ + "Error.log" )
```

JP1/Script version

Supported from JP1/Script 10-00.

9.9.5 ResetTrialOpenMode (cancel the trial-open function)

Purpose

Cancels the trial-open function.

Syntax

```
ResetTrialOpenMode
```

This command has no arguments.

Description

The `ResetTrialOpenMode` command cancels the trial-open operation to be performed if a lock error occurs during attempt to open a file that was closed by the `TextClose` command.

Example

```
Dim file1, Buff1
file1 = TextOpen( _BIN_ + "Error.log", ReadWrite )
If file1 = 0 Then
    MsgBox( _BIN_ + "Failed to open Error.log", OK )
    Exit 2
End If
TextSeek( file1, ToEnd )
TextWrite( file1, "A lock error occurred." )
SetTrialOpenMode( 2, 6000 ) ' Set trial-open.
TextClose( file1 ) ' Disk write might delay
ResetTrialOpenMode ' Cancel trial-open.
Exec( "MessageOutput.EXE", True, _BIN_ + "Error.log" )
```

JP1/Script version

Supported from JP1/Script 10-00.

10

Script Control Interface

This chapter describes the script control interface (API) for controlling a script during execution.

10.1 About the script control interface

A script control interface (API) controls scripts being executed. This interface is included in `SPTHL.DLL` supplied as part of JP1/Script, and can be called from another program by using the `LoadLibraryEx` function of the Windows API.

When calling the script control interface, specify `LOAD_WITH_ALTERED_SEARCH_PATH` for the execution flag of the entry point.

10.2 List of script control functions

The following script control functions are provided in JP1/Script:

Function	Meaning
SPTHOpen	Open the script control manager.
SPTHClose	Close the script control manager.
SPTHGetErrorMessage	Get an error message.
SPTHTerminate	Forcibly terminate script execution.

10.2.1 SPTHOpen (open the script control manager)

Purpose

Opens the script control manager, a program that controls the script execution process.

Prototype declaration

```
BOOL  APIENTRY  SPTHOpen
(
    LPCSTR      lpzComputerName,
    HWND        hWnd,
    LPHANDLE    lphScript
);
```

Arguments

lpzComputerName

Write the computer name of the script.

If you specify NULL, the current computer name is assumed.

hWnd

Specify the window handle for the calling process.

lphScript

Specify a pointer to the area in which to store the script control handle.

Return values

The function returns TRUE on successful execution, or FALSE if an error occurs. To get extended error information, use the GetLastError function.

JP1/Script version

Supported from JP1/Script 05-00.

10.2.2 SPTHClose (close the script control manager)

Purpose

Closes the script control manager.

Prototype declaration

```
void APIENTRY SPTHClose
(
    HANDLE hScript
);
```

Argument

hScript

Specify the script control handle returned by the `SPTHOpen` function.

Return values

This function has no return values.

JP1/Script version

Supported from JP1/Script 05-00.

10.2.3 SPTHGetErrorMessage (get an error message)

Purpose

Gets an error message.

Prototype declaration

```
DWORD APIENTRY SPTHGetErrorMessage
(
    DWORD dwErrorCode,
    LPSTR lpszMessage,
    DWORD dwSize
);
```

Arguments

dwErrorCode

Specify the error code obtained by the `GetLastError` function.

lpszMessage

Specify a pointer to the buffer in which to store the error message.

dwSize

Specify the byte size (including `\0`) of the buffer in which to store the error message. If the specified size is too small, the message will be truncated.

Return values

Error message (excluding `\0`).

JP1/Script version

Supported from JP1/Script 05-00.

10.2.4 SPTHTerminate (forcibly terminate script execution)

Purpose

Forcibly terminates script execution.

Prototype declaration

```
BOOL WINAPI SPTHTerminate
(
    HANDLE hScript,
    LPCSTR lpszFileName,
    UINT uProcessID,
    DWORD dwOption
);
```

Arguments

hScript

Specify the script control handle returned by the `SPTHOpen` function.

lpszFileName

Specify the full path of the script file.

This value must be the full path from the computer specified in the `SPTHOpen` function.

If you specify *uProcessID*, you can omit this value (specify `NULL`).

uProcessID

Specify the process ID of the script file.

If you specify *lpszFileName*, you can omit this value (specify `0`).

dwOption

Specify `0` or the following value:

Value	Meaning
<code>SPTH_TERM_CHILD</code>	Executable files called by the script from the <code>Exec</code> or <code>NetExec</code> command are also terminated. If the executable files do not terminate within three minutes after the request, the <code>TerminateProcess</code> function of the Win32 API forcibly terminates them.
<code>SPTH_DQ_FILENAME</code>	If the value specified in <i>lpszFileName</i> includes spaces and is enclosed by double quotation marks, the spaces are identified as part of the file name. If the value is not enclosed by double quotation marks, the string after the last space is identified as the file name.

Description

The `SPTHTerminate` function returns `TRUE` on successful execution, or `FALSE` if an error occurs. To get extended error information, use the `GetLastError` function.

This function ends normally (returns `TRUE`) even if there is no process to be terminated (the process has already terminated).

If this function ends normally, you can tell whether the process was actually terminated by referencing the return value of the `GetLastError` function.

If the return value is `NO_ERROR`, the process was forcibly terminated. Any other return value means that termination processing was not performed.

Note

Although you can omit either *lpszFileName* or *uProcessID*, both arguments should be specified whenever possible to identify the process to be terminated with greater certainty.

You must specify *uProcessID* to terminate a script process activated concurrently from multiple programs.

The return value is set in the following registry when you forcibly terminate script execution using the `SPTHTerminate` function. The default exit code is `17`.

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\ExitCode

Value name

Terminate

Value data type

REG_DWORD

Value

Return value of the script process

When the setting takes effect

The setting takes effect the next time the script file is executed.

Supplement

SPTHTerminate is used to forcibly terminate a script from JP1/AJS. For details, see [2.7.2\(6\) Forcibly terminating JP1/Script from JP1/AJS](#).

JP1/Script version

Supported from JP1/Script 05-00.

10.3 Coding example of a script control interface

A coding example of a script control interface is shown below.

Example

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include "sthapi.h"

BOOL TerminateScript(LPCSTR lpszNetName, LPCSTR lpszFileName, UINT
uProcessID)
{
    HKEY          hKey;
    LPCSTR        subKey =
        "SOFTWARE\\Hitachi\\JP1/Script\\PathName";

    char          valueName[32];
    char          value[_MAX_PATH];
    DWORD         valueSize = sizeof(value);
    HINSTANCE     hAPIInstance;
    tSPTHOpen     pSPTHOpen;
    tSPTHClose    pSPTHClose;
    tSPTHTerminate pSPTHTerminate;
    HANDLE        hScript;

    // Open the registry.
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE, subKey, 0,
        KEY_QUERY_VALUE, &hKey) != ERROR_SUCCESS)
    {
        // Error processing
        return(FALSE);
    }

    // Get the "SPTHL.DLL" path.
    strcpy(valueName, "Path01");
    if(RegQueryValueEx(hKey, valueName, 0, NULL, (LPBYTE) value,
&valueSize) != ERROR_SUCCESS)
    {
        // Error processing
        return(FALSE);
    }

    // Load "SPTHL.DLL".
    strcat(value, "\\SPTHL.DLL");
    hAPIInstance = LoadLibraryEx(value, NULL,
LOAD_WITH_ALTERED_SEARCH_PATH);
    if(hAPIInstance == NULL)
    {
        // Error processing
        return(FALSE);
    }

    // Get the entry point of each function.
    pSPTHOpen = (tSPTHOpen) GetProcAddress(hAPIInstance, "SPTHOpen");
    pSPTHClose = (tSPTHClose) GetProcAddress(hAPIInstance, "SPTHClose");
    pSPTHTerminate = (tSPTHTerminate) GetProcAddress(hAPIInstance,
```

```

"SPTHTerminate");

    // Open DLL.
    if (pSPTHOpen(lpszNetName, NULL, &hScript) == FALSE)
    {
        FreeLibrary(hAPIInstance);
        return(FALSE);
    }

    // Forcibly terminate script execution.
    if (pSPTHTerminate(hScript, lpszFileName, uProcessID, SPTH_TERM_CHILD)
== FALSE)
    {
        pSPTHClose(hScript);
        FreeLibrary(hAPIInstance);
        return(FALSE);
    }

    // Close DLL.
    pSPTHClose(hScript);

    // Release the API interface handle.
    FreeLibrary(hAPIInstance);

    return(TRUE);
}

```


11

Script OLE Control

This chapter describes the script OLE control for executing script commands.

11.1 About the script OLE control

The script OLE control is for executing script commands. This OLE control (SPTO.OCX) is supplied as part of JP1/Script and can be included in Visual Basic or other applications.

The script OLE control is supported in JP1/Script 05-00 and later versions.

11.2 SPTO control

Purpose

The SPTO control enables execution of a script command.

Syntax

```
SPTO
```

Properties

```
RTN  
EXECRTN  
EXECID  
EnableErrorMessage
```

Methods

```
SetGV  
GetGV  
DeleteGV  
Exec
```

JP1/Script version

Supported from JP1/Script 05-00.

11.2.1 RTN property (return the error detail code for a method)

Purpose

Returns the error detail code for a method.

Syntax

```
Object.RTN
```

Argument

Object

Specify the object representing the reference to the SPTO control.

Description

The RTN property returns the error detail code of a method as a long integer value.

JP1/Script version

Supported from JP1/Script 05-00.

11.2.2 EXECRTN property (return the exit code of an executable file)

Purpose

Returns the exit code of an executable file called by the Exec method (with True set in the wait flag).

Syntax

```
Object.EXECRTN
```

Argument

Object

Specify the object representing the reference to the SPTO control.

Description

The EXECRTN property waits for completion of an application called by the Exec method, and stores the application's exit code as a long integer if the Exec method returns a True result. Nothing is stored if the method's execution result is False, or if the Exec method is called with False set in the wait flag.

JP1/Script version

Supported from JP1/Script 05-00.

11.2.3 EXECID property (return the identifier of an executable file)

Purpose

Returns the identifier of an executable file called by the Exec method (with False set in the wait flag).

Syntax

```
Object.EXECID
```

Argument

Object

Specify the object representing the reference to the SPTO control.

Description

The EXECID property stores the executable file ID as a string value when an application is called by the Exec method with False set in the wait flag.

JP1/Script version

Supported from JP1/Script 05-00.

11.2.4 EnableErrorMessage property (check for errors in the SPTO control)

Purpose

Checks whether the SPTO control contains any errors.

Syntax

```
Object.EnableErrorMessage
```

Argument

Object

Specify the object representing the reference to the SPTO control.

Description

The `EnableErrorMessage` property returns `False` if no errors are found in the SPTO control, or `True` otherwise.

JP1/Script version

Supported from JP1/Script 05-00.

11.2.5 SetGV method (set a global variable)

Purpose

Sets a global variable.

Syntax

```
Object.SetGV(GlobalName, Value)
```

Arguments

Object

Specify the object representing the reference to the SPTO control.

GlobalName

Specify the global variable to set as a string value.

Value

Specify the value as a string value.

Description

The `SetGV` method sets a specified value in a specified global variable. The method returns `True` on successful execution, or `False` if an error occurs.

If you specify a non-existent global variable, a new global variable is created with the specified value. If you specify an existing global variable, its value is updated.

Note

This method creates a global variable file (`SPTGV.SPG`) in the `DATA` folder[#] in the installation folder. However, in JP1/Script 06-51 and later versions, if you open the Options (Cluster Environment) dialog box (**Tools, Options** command) and change the folder for management file output, the file will be created in the folder you specified.

The global variable file is preserved when the script completes execution. Delete this file if you want to initialize the global variables.

#

The file is created in the script execution environment folder (`system-drive\ProgramData\Hitachi\Script\Data`).

JP1/Script version

Supported from JP1/Script 05-00.

11.2.6 GetGV method (get a global variable)

Purpose

Gets a global variable.

Syntax

```
Object.GetGV(GlobalName)
```

Arguments

Object

Specify the object representing the reference to the SPTO control.

GlobalName

Specify the global variable to be acquired as a string value.

Description

The `GetGV` method acquires a specified global variable and returns its value. If you specify a non-existent global variable, a zero-length string ("") is returned.

JP1/Script version

Supported from JP1/Script 05-00.

11.2.7 DeleteGV method (delete a global variable)

Purpose

Deletes a global variable.

Syntax

```
Object.DeleteGV(GlobalName)
```

Arguments

Object

Specify the object representing the reference to the SPTO control.

GlobalName

Specify the global variable to be deleted as a string value. Or, you can specify the following value:

Value	Meaning
AllGV	Delete all global variables, but preserve the global variable file (SPTGV.SPG).

Description

The `DeleteGV` method deletes a specified global variable. The method returns `True` on successful execution, or `False` if an error occurs.

If you specify a non-existent global variable, the method returns `True` without doing anything.

JP1/Script version

Supported from JP1/Script 06-51.

11.2.8 Exec method (call an executable file)

Purpose

Calls an executable file (EXE file, BAT file, COM file, SPT file, CMD file, or linked file) according to specified parameters. You can specify whether to wait for the called application to complete.

Syntax

```
Object.Exec(FileName, Flag, Parameter)
```

Arguments

Object

Specify the object representing the reference to the SPTO control.

FileName

Specify the executable file to be called as a string value.

You can specify any of the following types of files:

- Executable file (.EXE)
- MS-DOS batch file (.BAT)
- MS-DOS executable file (.COM)
- JP1/Script script file (.SPT)
- command script (.CMD)
- Linked file

Flag

Specify a Boolean value indicating whether to wait for the executable file specified in *FileName* to complete execution. Specify `True` to wait for completion; otherwise, specify `False`.

Parameter

Specify the necessary parameters for executing the file specified in *FileName*. Write each parameter as a string value, using a space to separate each parameter.

The following strings have a special meaning when specified as a parameter:

Parameter	Meaning
/SPT:HIDE	Hides the application window.
/SPT:MIN	Minimizes the application window to an icon.
/SPT:MAX	Maximizes the application window.

The parameters are passed as command line parameters to the executable file. The strings `/SPT:HIDE`, `/SPT:MIN`, and `/SPT:MAX` are not passed as command line parameters.

Description

The `Exec` method executes a specified executable file.

If you specify `True` in *Flag*, execution control waits for the called application to complete. If you specify `False`, control moves to the next processing without waiting for the application to complete.

The method returns `True` if the application completes normally, or `False` in all other cases.

If you specify `True` in *Flag* and the execution result is `True`, the executable file's exit code is stored as a long integer in the `EXECRTN` property. Nothing is stored in this property if you specify `False` in *Flag* or if `False` is returned as the method's execution result.

If you specify `False` in *Flag*, the executable file ID is stored as a string value in the `EXECID` property.

JP1/Script version

Supported from JP1/Script 05-00.

Appendixes

A. Output Formats of Script Trace Files

This appendix describes the output formats of the following Script trace files:

- Analysis trace file
- Execution trace file
- User trace file (trace file)
- Server trace file
- NetExec error log file

A.1 Output formats of analysis trace files

An analysis trace file is used to store command analysis results. The output format depends on whether the script can be activated concurrently from multiple processes.

(1) Multi-activation prohibited

- File name
The file name (extension `.SPA`) is a combination of the folder name and the script file name specified in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.
- Output format
Figure A-1 shows the output format of an analysis trace file.

Figure A–1: Output format of an analysis trace file (multi-activation prohibited)

```
2003/02/07 14:23:58 << ANALYSIS RESULT - SOP3F370-2000 / Administrator >>-----0700
a
 14:23:58      7.9      : Message      : Required argument is missing.
b              c        d              e
TOTAL ERROR    1
f
2003/02/07 14:23:58 -----
g
2003/02/07 14:25:00 << ANALYSIS RESULT - SOP3F370-2000 / Guest >>-----0700

TOTAL ERROR    0

2003/02/07 14:26:01 -----
```

- a. Date and time at which trace output started, the computer name and user name that executed the script, and the Script Engine version
- b. Time at which an analytical error occurred
- c. Line and column positions at which an analytical error occurred
- d. Name of the command in which an analytical error occurred

- e. Type of analytical error
 - f. Total number of analytical errors
 - g. Date and time at which trace output ceased
- Maximum lines and columns
Trace information can be output to an analysis trace file to a maximum of between 100 and 9,999 lines and 128 to 1,024 columns.
You can change these output limits in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.
If the output trace information exceeds the maximum line count, Script Engine returns to the first line and outputs the new information, replacing the existing information. If the maximum column count is exceeded, the excess columns are deleted.

(2) Multi-activation permitted

- File name
The file name (extension .SPA) is a combination of the folder name and the script file name specified in the Options (Multi-activation) dialog box, accessed by choosing **Tools, Options** in the Script Manager window.
- Output format
Figure A-2 shows the output format of an analysis trace file.

Figure A–2: Output format of an analysis trace file (multi-activation permitted)

```

2003/02/07 14:56:14 149      S 0700 - SOP3F370-2000 / Administrator
a
2003/02/07 14:56:14 149          5.9      : NetExec          : Insufficient arguments.
b          c          d          e          f
2003/02/07 14:56:14 2244      S 0700 - SOP3F370-2000 / Guest
2003/02/07 14:56:14 2244          5.9      : NetExec          : Insufficient arguments.
2003/02/07 14:56:17 149          8.9      : Message          : Required argument is
missing.
2003/02/07 14:56:17 149      E          TOTAL ERROR      2
g          h
2003/02/07 14:56:17 2244          8.9      : Message          : Required argument is
missing.
2003/02/07 14:56:17 2244      E          TOTAL ERROR      2

```

- a. Date and time at which trace output started for the executed process, the process identifier, letter S indicating the start, the Script Engine version, and the computer name and user name that executed the process
- b. Time at which an analytical error occurred
- c. Process identifier of the process in which an analytical error occurred
- d. Line and column positions at which an analytical error occurred
- e. Name of the command in which an analytical error occurred

- f. Type of analytical error
- g. Date and time at which trace output ceased for the executed process, the process identifier, and the letter E indicating the end.
- h. Total number of analytical errors

- Maximum lines and columns

Trace information can be output to an analysis trace file to a maximum of between 100 and 9,999 lines and 128 to 1,024 columns.

You can change these output limits in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.

If the output trace information exceeds the maximum line count, Script Engine returns to the first line and outputs the new information, replacing the existing information. If the maximum column count is exceeded, the excess columns are deleted.

A.2 Output formats of execution trace files

An execution trace file contains the command execution results output by the Script Execution program. The output format depends on whether the script can be activated concurrently from multiple processes.

(1) Multi-activation prohibited

- File name

The file name (extension .SPX) is a combination of the folder name and the script file name specified in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.

- Output format

Figure A-3 shows the output format of an execution trace file.

Figure A-3: Output format of an execution trace file (multi-activation prohibited)

```

2003/02/07 14:23:58 << EXECUTION RESULT - SOP3F370-2000 / Administrator >>----- 0700
a
    * TOTAL ERROR      0
    Syntax errors found. See the analysis result.
2003/02/07 14:23:58 -----
2003/02/07 14:25:00 << EXECUTION RESULT - SOP3F370-2000 / Guest >>----- 0700
14:24:23 *      11: DeleteFile      : 0032:Unable to access the file.
                                Another process is using the file. (d:\Script\job\
job004.txt)
14:24:27 *      13: MakeDir       : Failed to make the specified directory. (A:\Script)
14:25:45 *      25: Exec          : 0002:Unable to find the specified file.
                                (D:\Script\Job\Job004.exe)
b      c      d      e      f
    * TOTAL ERROR      3
    g
2003/02/07 14:26:01 -----
h

```

- a. Date and time at which trace output started, the computer name and user name that executed the script, and the Script Engine version

- b. Time at which an execution error occurred
 - c. Error indicator
 - d. Line position of the command in which an execution error occurred
 - e. Name of the command or procedure in which an execution error occurred
 - f. Type of error
 - g. Total number of execution errors
 - h. Date and time at which trace output ceased
- Maximum lines and columns
Trace information can be output to an execution trace file to a maximum of between 100 and 9,999 lines and 128 to 1,024 columns.
You can change these output limits in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.
If the output trace information exceeds the maximum line count, the Script Execution program returns to the first line and outputs the new information, replacing the existing information. If the maximum column count is exceeded, the excess columns are deleted.

(2) Multi-activation permitted

- File name
The file name (extension `.SPX`) is a combination of the folder name and the script file name specified in the Options (Multi-activation) dialog box, accessed by choosing **Tools, Options** in the Script Manager window.
- Output format
Figure A-4 shows the output format of an execution trace file.

Figure A–4: Output format of an execution trace file (multi-activation permitted)

```

2002/02/07 15:08:46 596      S 0671 - SOP3F370-2000 / Administrator
a
2002/02/07 15:08:47 596      *      5: Message      : 0123:Incorrect syntax in file
name, directory name, or volume label.
      b      c      d      e      f      g
2002/02/07 15:08:47 2296    S 0671 - SOP3F370-2000 / Guest
2002/02/07 15:08:47 2296    *      5: Message      : 0123:Incorrect syntax in file
name, directory name, or volume label.
2002/02/07 15:08:48 596      *      7: MakeDir      : Failed to make the specified
directory. (A:\Script)
2002/02/07 15:08:48 2296    *      7: MakeDir      : Failed to make the specified
directory. (A:\Script)
2002/02/07 15:08:48 596      *      10: Exec       : 0002:Unable to find the specified
file. (D:\Script\job\job004.exe)
2002/02/07 15:08:48 2296    *      10: Exec       : 0002:Unable to find the specified
file. (D:\Script\job\job004.exe)
2002/02/07 15:10:39 596      *      13: SetGV      : 0053:Unable to find the network
path. (SOP3F370-NT)
2002/02/07 15:10:39 596      E *      TOTAL ERROR      4
h
2002/02/07 15:10:44 2296    *      13: SetGV      : 0053:Unable to find the network
path. (SOP3F370-NT)
2002/02/07 15:10:44 2296    E *      TOTAL ERROR      4

```

- a. Date and time at which trace output started for an executed process, the process identifier, letter S indicating the start, the Script Engine version, and the computer name and user name that executed the process
- b. Time at which an execution error occurred
- c. Process identifier of the process in which an execution error occurred
- d. Error indicator
- e. Line position of the command in which an execution error occurred
- f. Name of the command or procedure in which an execution error occurred
- g. Type of error
- h. Date and time at which trace output ceased for an executed process, the process identifier, and the letter E indicating the end.
- i. Total number of execution errors

- Maximum lines and columns
Trace information can be output to an execution trace file to a maximum of between 100 and 9,999 lines and 128 to 1,024 columns.

You can change these output limits in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.

If the output trace information exceeds the maximum line count, Script Execution returns to the first line and outputs the new information, replacing the existing information. If the maximum column count is exceeded, the excess columns are deleted.

A.3 Output format of user trace files

A user trace file is output when `Target_File` is specified in the first argument of the `Message` command.

(1) File name

The file name (extension `.TXT`) can be any name assigned by the user.

(2) Output format

Figure A-5 shows the output format of a user trace file.

Figure A-5: Output format of a user trace file

```
2002/02/24 12:00:01 Start
2002/02/24 12:01:10 Job001      Job001 called
2002/02/24 12:01:15 Job001      Job001 call completed (normal)
2002/02/24 12:03:17 Script execution completed
a                               b
```

- a. Date and time at which a trace was output
- b. Output message

(3) Maximum lines and columns

Trace information can be output to a user trace file to a maximum of between 100 and 9,999 lines and 128 to 1,024 columns.

You can change these output limits in the Trace Information page, accessed by choosing **File, Set Execution Environment** in the Script Manager window.

If the output trace information exceeds the maximum line count, the system returns to the first line and outputs the new information, replacing the existing information. If the maximum column count is exceeded, the excess columns are deleted.

A.4 Output format of server trace files

A server trace file contains the results of command execution by the server when called from the `SetGV`, `GetGV`, `DeleteGV`, or `NetExec` command written in a script running on a client.

(1) File name

The file name is fixed as `SPTSVTRC.SPY`. The file is created in the `DATA` folder[#] in the installation folder.

- # The file is created in the script execution environment folder (`system-drive\ProgramData\Hitachi\Script\Data`).

(2) Output format

Figure A-6 shows the output format of the server trace file.

Figure A–6: Output format of the server trace file

2002/02/19	22:19:03	22:19:03	SetGV			OK	"A65(T)"	"TAK"	"D:\SCR\VAR\N_SGV.SPT(23)"
2002/02/19	22:19:15	22:19:15	GetGV			ER(2)	"A65(T)"	"TAK"	"D:\SCR\VAR\N_GGV.SPT(21)"
							g		
2002/02/19	22:20:16	22:20:22	NetExec	S	151	OK	"A65(T)"	"TAK"	"D:\SCR\EXEC\N_PPM.SPT(29)"
a	b	c	d	e	f	g	h	i	j
2002/02/19	22:20:16	22:20:22	NetExec	E	151	0			
	k	l		e		m			

- a. Date on which the server executed the command
- b. Time at which the server started command execution
- c. Time at which the server completed command execution
- d. Name of the executed command
- e. When the executed command calls a process, a two-line trace is produced. A symbol identifies whether the output line is the first or second line of the trace.
 - S: First line (start)
 - E: Second line (end)
- f. Process identifier output when the executed command calls a process
- g. Execution result of the command executed by the server
 - OK: Successful
 - ER: Execution error (error detail code in parentheses)
- h. Computer name of the client (enclosed with double quotes)
- i. User name of the client (enclosed with double quotes)
- j. Full path of the script file executed by the client (enclosed with double quotes). The number in parentheses is the line number in the script file of the command executed by the server.
- k. Start time of a process called by the server that executed the command
- l. End time of a process called by the server that executed the command

m.

Completion code of a process called by a command executed on the server (output only if the command execution result is OK)

(3) Maximum lines and columns

Execution results are output to a server trace file up to a maximum of 1,000 lines and 200 columns. These limits cannot be changed.

If the output execution results exceed the maximum line count, the system returns to the first line and outputs the new results, replacing the existing results. If the maximum column count is exceeded, the results continue onto a new line.

(4) Output specification

You can specify whether or not to output a server trace file in the Server Information page of the Options dialog box which opens from the Manager window.

A.5 Output format of NetExec error log files

A NetExec error log file contains information about errors that occur during execution of the NetExec command. If you cannot determine the cause of an error by referring to the trace file, log file, and event log, acquire a NetExec log file and then contact the support center.

(1) File name

- Client system

The file name is `STXNetExec_C_XX.LOG`^{#1}.

This file is output to the `STXNetExec_Client` folder that is created in the LOG folder^{#2, 3} in the installation folder.

- Server system

The file name is `STXNetExec_S_XX.LOG`^{#1}.

This file is output to the `STXNetExec_Server` folder that is created in the LOG folder^{#2, 4} in the installation folder.

#1

`XX` is a numeric value in the range 01 to 30. One trace file is output for each execution of a process of `SPTXNetx.exe` (which controls NetExec command execution).

#2

The folder is created in the script log file folder (`system-drive\ProgramData\Hitachi\Script\Log`).

#3

In versions earlier than JP1/Script 07-00, the folder is created in `installation-folder\Data\LOG`.

#4

In versions earlier than JP1/Script 07-00, the folder is created in `installation-folder\Data\LOG`.

(2) Output format

Figure A-7 shows the output format of the error log in the client system, and Figure A-8 shows the output format of the error log in the server system.

Figure A-7: Output format of the error log in the client system

```
11:44:57:659 2003-02-07 *** CallCompName=SOP3F370-2000, CallUserName=WADA
11:44:57:659 2003-02-07 CallFileName=D:\TEST_005.SPT,CallFileLine=0
11:44:57:659 2003-02-07 * NetExec->CompName=10.210.104.103, FileName=D:\TEST\MgBox.spt
11:44:57:659 2003-02-07 Flag=1,ExecDirName=
ExecPlace=0,Option=1325
11:44:57:574 2003-02-07 <<< CxNamePipeClient::Connect Error End 2 (RC=2)
11:44:57:850 2003-02-07 <<< CxNamePipeBasic::SendData Error End 4 (RC=1431698578)
11:44:57:850 2003-02-07 <<< STXDNetworkProcessClient::CheckGetResultNetExec Error End 1
(RC=1431698578)
11:44:57:850 2003-02-07 <<< STXDNetExecApp::Exitinstance End (ExitCode=-1431698578)
```

Figure A-8: Output format of the error log in the server system

```
0035653767 0.000 19:02:15:087 2003-02-07 >>> CxNamedPipeServer::Create -> PipeName=JP1
Script-NetworkExec-ByLogon, TimeOut=60000
0035653767 0.000 19:02:15:187 2003-02-07 <<< STXDNetworkProcessServer::OnRequestDoNet
Exec Error End 1
```

(3) File size

The size of an error log file for both client and server systems is as follows:

Maximum size of one file: (100 bytes/line) * 100 lines = approximately 10 kilobytes

Maximum number of files: 30

Maximum value: Approximately 10 kilobytes + 30 files = approximately 300 kilobytes

(4) Output specification

As the default, the installer sets the following registry and outputs the error log:

Registry key

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\SPTX\Option
```

Value name 1

```
Net_Trace
```

Value datatype 1

```
REG_DWORD
```

Value 1

Whether or not to output trace and its output level:

0: Do not output trace.

1: Output only error trace and delete the file at normal termination (this is the default value).

When the setting takes effect

The setting takes effect the next time the script file is executed.

Value name 2

Net_Trace_Directory

Value datatype 2

REG_SZ

Value 2

Trace output destination folder name (initial value: "script log folder")

When the setting takes effect

The setting takes effect the next time the script file is executed.

B. Sample Files

JP1/Script provides two sample files:

- A sample file that executes processing according to numbers selected in a displayed menu
- A sample file that calls a program and checks for a program completion code

This appendix outlines the functions and organization of these sample files and describes how to run them. For details about the installation folder for the sample files, see [2.1.1 Program installation folders](#).

B.1 Sample file that executes processing according to numbers selected in a displayed menu

(1) Summary of functions

When you execute the sample file, a menu appears. The following actions are performed by selecting numbers in the menu:

Process No. 1

Creates a script file.

Process No. 2

Copies the script file.

Process No. 3

Outputs a list of files in the folder that contains the sample file to the user trace file.

Process No. 9

Terminates the sample file.

(2) File organization

`SmplMain.Spt`: Main script file of this sample file

`Smpl001.Spt`: Script file for process No. 2

`Smpl002.Spt`: Script file for process No. 3

`Sample.Spt`: Default script file created in process No. 1.

(3) Executing the sample file

1. Execute the script file `SmplMain.Spt`.

The menu window appears.

2. Type the number of the process that you want to execute. Then choose the **Run** button. If you choose **Cancel**, the sample program ends.

- Select **1** to open the window for specifying the script file name. Enter the file name, and then click the **OK** button.

The specified script file opens.

- Select **2** to copy the script file created in process No. 1. A folder named `CopyFolder` is created in the folder containing the sample file, and the file is copied with the file name `bkupSpt.spt`.
- Select **3** to output a list of files in the folder containing the sample file to the user trace file. The name of the user trace file is `SmplOut.TXT`.

When processing terminates normally, the display returns to the menu window.

If an error occurred, a beep sounds, and a message box opens. In addition, a user error warning is output as the status information in Trace Viewer.

B.2 Sample file that calls an executable file and checks for a program completion code

(1) Summary of functions

This sample file calls an executable file and checks the exit code.

(2) File organization

`ExitCode.SPT`: Main script file of this sample file

`ExitCode.EXE`: Executable file that is called by `ExitCode.Spt`

(3) Executing the sample file

1. Execute the sample file `ExitCode.SPT`.

Specify either of the following values as a parameter:

T: Execute the sample file in interactive mode.

B: Execute the sample file in batch mode.

2. If you execute the sample file in interactive mode, a window appears. Enter the exit code.

If you enter a value other than 0 for the exit code, a user trace file named `EXITCODE.TXT` is output.

If you execute the sample file in batch mode, the window does not appear and exit code 0 is assumed.

If an error occurs, a message box opens and processing terminates.

C. Error Detail Codes

C.1 Error codes set by JP1/Script

The following table describes the error codes that JP1/Script sets in `_RTN_` after the `NetExec` command has executed.

Error code	Error description
536904787	A server is not logged on with the logon user for the start parameter of the Script Launcher service.
-1431698655	A timeout error occurred while JP1/Script was waiting for a response from the server.
-1431698656	The server is not responding to the connection request.
-1431698576	The user for executing the <code>NetExec</code> command is not permitted to execute the command on the called computer.
-1431698579	An error occurred while data was being received from the server.
-1431698578	During communication with the server, the allowed non-communication time was exceeded.

C.2 OS error codes

The following table lists the error codes set by Windows. This information is provided for reference only; the contents are not guaranteed because the values may differ depending on the OS.

Error code	Error description
-004	The memory space for output files is insufficient.
-006	A global handle is incorrect.
0001	A function is incorrect.
0002	The system cannot find a specified file.
0003	The system cannot find a specified network name or directory path.
0004	A file cannot be opened.
0005	Access was denied. Recheck file attributes or security.
0006	A handle is invalid.
0007	A storage control block has been destroyed.
0008	There is insufficient storage to execute this command.
0009	The address of a storage control block is invalid.
0010	The environment is incorrect.
0011	A program with an incorrect format was read.
0012	An access code is invalid.
0013	Data is invalid.
0014	There is insufficient storage to complete this operation.
0015	A specified drive cannot be found.
0016	A directory cannot be deleted.

Error code	Error description
0017	A file cannot be moved to another directory.
0018	There is no higher file.
0019	A medium is write-protected.
0020	A specified device cannot be found.
0021	A device is not ready.
0022	A device cannot recognize the command.
0023	A data error (cyclic redundancy check (CRC) error) was detected.
0024	The program issued a command, but the command length is incorrect.
0025	A specified disk area or track cannot be found.
0026	A specified disk or floppy disk cannot be accessed.
0027	A requested sector cannot be found.
0029	Cannot write to a specified device.
0030	Cannot read from a specified device.
0031	A device connected to the system is not functioning.
0032	A process cannot access a file because another process is using the file.
0033	A process cannot access a file because another process has locked part of the file.
0036	Too many shared files are open.
0038	The end of a file was reached.
0039	A disk is full.
0050	Network requests are not supported.
0051	A remote computer cannot be used.
0052	The same name already exists in the network.
0053	The network path cannot be found.
0054	The network is busy.
0055	A specified network resource or device cannot be used.
0056	The network BIOS command reached the limit.
0057	A hardware error occurred in the network adapter.
0058	A specified server cannot execute the requested operation.
0059	An unexpected network error occurred.
0060	The remove adapter is not compatible.
0064	The specified network name cannot be used.
0065	Network access was rejected.
0066	The network resource type is incorrect.
0067	The network name cannot be found.
0068	The number of names for network adapter cards in the local computer exceeded the limit.

Error code	Error description
0069	The number of network BIOS sessions exceeded the limit.
0070	The remote server is temporarily stopped or is being started.
0071	Another remote computer cannot be connected because the maximum number of connections to computers has been reached.
0072	A specified printer or disk device has stopped temporarily.
0080	A file already exists.
0082	A directory or file cannot be created.
0084	There is insufficient storage to process this request.
0085	A local device name is already being used.
0086	The specified network password is incorrect.
0087	A parameter is incorrect.
0088	A write error occurred in the network.
0089	Another process cannot be started at this time.
0110	A specified device or file cannot be opened.
0111	A file name is too long.
0112	A disk does not have enough free space.
0123	The syntax of a file name, directory name, or volume name is incorrect.
0144	A directory is not a subdirectory of the root directory.
0145	A directory is not empty.
0148	A specified path cannot be used at this time.
0161	A specified path is invalid.
0164	This system cannot create any more threads.
0167	A file area cannot be locked.
0170	A requested resource is being used.
0183	A file that already exists cannot be created.
0196	This application program cannot be executed on this operating system.
0197	The current configuration of the operating system does not allow execution of this application program.
0199	This application program cannot be executed on this operating system.
0206	A file name or extension is too long.
0230	A pipe status is disabled.
0231	All pipe instances are busy.
0232	A pipe is closed.
0240	The session was canceled.
0267	A directory name is invalid.
1053	This service did not respond to the startup or control request within the specified time.
1054	A thread cannot be created for the service.

Error code	Error description
1056	A service instance is already being executed.
1057	The account name is invalid or does not exist, or the password for the specified account name is invalid.
1058	The specified service cannot be started because the service is invalid or is not associated with a valid device.
1060	The specified service does not exist as an installed service.
1062	This service cannot be started.
1069	The service cannot be started because logon failed.
1070	The service froze in start wait status after being started.
1073	The specified service has already been started.
1078	This name is already being used as a service name or a service display name.
1114	Execution of the dynamic link library (DLL) initialization routine failed.
1115	System shutdown is being executed.
1117	The request cannot be executed because an I/O device error occurred.
1123	The sector ID field of the floppy disk and the track address of the floppy disk controller track do not match.
1124	The floppy disk controller reported an error in which the floppy disk driver is not recognized.
1125	The floppy disk controller returned an inconsistent result to the register.
1130	The server cannot allocate a storage area required for processing this command.
1131	The possibility of a deadlock occurrence was detected.
1176	The replacing file cannot be moved to the file to be replaced. The name of the file to be replaced remains the same.
1177	The replacing file cannot be moved to the file to be replaced. The name of the file to be replaced has been changed to the backup name.
1326	Logon failure: The user name cannot be recognized, or the password is incorrect.
1327	Logon failure: User account restriction
1328	Logon failure: Constraint violation of the account logon time
1329	Logon failure: The user is not allowed to log on to this computer.
1330	Logon failure: The effective period of the specified account and password has expired.
1331	Logon failure: The account is currently disabled.
1380	Logon failure: This computer does not allow users to use the requested type of logon.
1385	Logon failure: This computer does not allow users to use the requested type of logon.
1450	The requested service cannot be completed because of insufficient system resources.
1451	The requested service cannot be completed because of insufficient system resources.
1452	The requested service cannot be completed because of insufficient system resources.
7006	A service with the same name already exists in the system.

You can use the `GetErrorMessage` command to obtain the error contents of the return code that is set after command execution. An example is shown below:

Example


```
Dim ErrMsg
Dim ErrCode
ErrCode=21
ErrMsg=GetErrorMessage(ErrCode)
Message(Target_File, _BIN_"ErrMsg.txt", ErrMsg)
```

D. Maintenance Log Files

A maintenance log file is output if an error occurs when a Windows function is called in a JP1/Script.

If an error occurs and you cannot determine the cause of the error by referring to the trace file, log file, or event log, acquire the maintenance log file and then contact the support center.

The following table lists the functions that output maintenance log files, and the log file names.

Table D–1: Functions that output maintenance log files and log file names

Log output function	Log file name
Script execution control	SPTXE_ <i>process-ID</i> .log
NetExec command execution control	SPTXNETX_ <i>process-ID</i> .log
Script service	SPTHSV_ <i>process-ID</i> .log
Script Launcher	SPTHLNCH_ <i>process-ID</i> .log
Script Launcher service	SPTHLV_ <i>process-ID</i> .log

If maintenance log files are repeatedly output by a single process, the output log file size might exceed the maximum allowable size. If the log file size exceeds the maximum, the name of the oldest log file will change to *XXXX_ process-ID . n . log* (where *n* is a serial number) the next time a maintenance log file is output. Then, a new *XXXX_ process-ID . log* file is created.

The following describes how to specify output of maintenance log files.

Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script\Option\Maintenance

When the setting takes effect

- For Script execution control, the next time the script file is executed
- For NetExec command execution control, the next time the script file is executed
- For the JP1/Script service, the next time the JP1/Script service is started
- For Script Launcher, the next time Script Launcher is started
- For the Script Launcher service, the next time the Script Launcher service is started

Table D–2: Values that can be set for registry keys

Function name	Value name	Value datatype	Value
Script execution control	SptExec_Log_Dir	REG_SZ	Output folder path ^{#1} Initial value: Default maintenance log file output folder ^{#2}
	SptExec_Log_Size	REG_DWORD	Maximum file size (bytes) 1,024 to 2,147,483,647 Initial value: 32,768
	SptExec_Log_Num	REG_DWORD	Maximum number of files 0 to 1,024 Initial value: 20

Function name	Value name	Value datatype	Value
NetExec command execution control	SptNetExec_Log_Dir	REG_SZ	Output folder path ^{#1} Initial value: Default Maintenance log file output folder ^{#2}
	SptNetExec_Log_Size	REG_DWORD	Maximum file size (bytes) 1,024 to 2,147,483,647 Initial value: 32,768
	SptNetExec_Log_Num	REG_DWORD	Maximum number of files 0 to 1,024 Initial value: 10
Script service	Service_Log_Dir	REG_SZ	Output folder path ^{#3} Initial value: Default Maintenance log file output folder ^{#2}
	Service_Log_Size	REG_DWORD	Maximum file size (bytes) 1,024 to 2,147,483,647 Initial value: 524,288
	Service_Log_Num	REG_DWORD	Maximum number of files 0 to 1,024 Initial value: 2
Script Launcher	Launcher_Log_Dir	REG_SZ	Output folder path ^{#4} Initial value: Default Maintenance log file output folder ^{#2}
	Launcher_Log_Size	REG_DWORD	Maximum file size (bytes) 1,024 to 2,147,483,647 Initial value: 524,288
	Launcher_Log_Num	REG_DWORD	Maximum number of files 0 to 1,024 Initial value: 2
Script Launcher service	LauncSv_Log_Dir	REG_SZ	Output folder path ^{#3} Initial value: Default Maintenance log file output folder ^{#2}
	LauncSv_Log_Size	REG_DWORD	Maximum file size (bytes) 1,024 to 2,147,483,647 Initial value: 524,288
	LauncSv_Log_Num	REG_DWORD	Maximum number of files 0 to 1,024 Initial value: 2

#1: Select a folder in which files can be created, deleted, and updated by using the account used for executing files.

#2: Maintenance log files are output to the following folder by default:

system-drive\ProgramData\Hitachi\Script\Log\Maintenance

#3: Select a folder in which files can be created, deleted, and updated by using a local system account.

#4: Select a folder in which files can be created, deleted, and updated by using the account used for starting files.

Generally, use the initial values of the maximum size of a maintenance log file and maximum number of maintenance log files. Maintenance log files might be output even for minor errors. Therefore, depending on the contents of a script, a maintenance log file might be created or updated even if the system is running normally. In such a case, change the

number of maintenance log files for script execution control or for the `NetExec` command. When changing the values, use the number of scripts that can be executed concurrently as a guideline.

If a version of JP1/Script earlier than 10-00 is used with a script that ignores errors and executes the next process, upgrading JP1/Script to 10-00 might degrade execution performance due to output of maintenance log files. If such performance degradation occurs, specify 0 for the maximum number of maintenance log files for script execution control. This will suppress output of maintenance log files.

E. Version Changes

This appendix describes the changes in each version.

E.1 Changes in version 11-00

- The JP1 call commands are deleted.
- The following OSs are now supported: Windows Server 2012, Windows 8, Windows 8.1, and Windows 10.
- The following OSs are no longer supported:
 - Windows XP
 - Windows Server 2003
 - Windows Vista
 - Windows Server 2008 (except Windows Server 2008 R2)

E.2 Changes in version 10-00

- The mode in which DLL files are not unloaded is now supported for the `CallDll` command.
- A function that restricts users who can remotely execute the `NetExec` command has been added.
- The Script Launcher service is now supported.
- RemoteApp (TS RemoteApp) of the Windows Server 2008 terminal service is now supported.
- For security warnings that have been output in Windows Vista or later versions of Windows, access permissions can now be set for files created by JP1/Script according to the user's operational needs.
- The trial-open function is now supported.
- The lock error retry function is now supported.
- Maintenance log files are now output.
- The following operating systems are no longer supported:
 - Microsoft(R) Windows Server(R) 2003, Enterprise Edition for Itanium-based Systems
 - Microsoft(R) Windows Server(R) 2008 for Itanium-based Systems
 - Microsoft(R) Windows Server(R) 2008 R2 for Itanium-based Systems

E.3 Changes in version 09-00

- Linkage with the JP1/AJS3 is now supported for system configurations used for automatically executing jobs and providing job execution control.
- The following has been deleted from the cases in which the `ExitWindows` command does not function: The command is executed from a service for which a non-system account has been set in Windows 2000 Server, Windows 2000 Advanced Server, or Windows Server 2003, and the service has not logged onto Windows.
- Notes on executing the `Beep` command have been added.
- Windows Server 2008 and Windows Server 2008 (IPF) are no longer supported.

F. Reference Material for This Manual

This appendix provides reference information, including various conventions, for this manual.

F.1 Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

About JP1/IM:

- *JP1 Version 11 Integrated Management: Getting Started* (3021-3-A06(E))
- *JP1 Version 11 JP1/Integrated Management - Manager Overview and System Design Guide* (3021-3-A07(E))
- *JP1 Version 11 JP1/Integrated Management - Manager Configuration Guide* (3021-3-A08(E))
- *JP1 Version 11 JP1/Integrated Management - Manager Administration Guide* (3021-3-A09(E))
- *JP1 Version 11 JP1/Integrated Management - Manager GUI Reference* (3021-3-A10(E))
- *JP1 Version 11 JP1/Integrated Management - Manager Command and Definition File Reference* (3021-3-A11(E))
- *JP1 Version 11 JP1/Integrated Management - Manager Messages* (3021-3-A12(E))

About JP1/AJS3:

- *JP1 Version 11 Job Management: Getting Started (Job Scheduler)* (3021-3-B11(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Overview* (3021-3-B12(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 System Design (Configuration) Guide* (3021-3-B13(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 System Design (Work Tasks) Guide* (3021-3-B14(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Configuration Guide* (3021-3-B15(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Administration Guide* (3021-3-B16(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Troubleshooting* (3021-3-B17(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Operator's Guide* (3021-3-B18(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Command Reference* (3021-3-B19(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Linkage Guide* (3021-3-B20(E))
- *JP1 Version 11 JP1/Automatic Job Management System 3 Messages* (3021-3-B21(E))

About JP1/AJS2:

- *Job Management Partner 1 Version 8 Job Management Partner 1/Automatic Job Management System 2 Description* (3020-3-K21(E))
- *Job Management Partner 1 Version 8 Job Management Partner 1/Automatic Job Management System 2 Planning and Administration Guide* (3020-3-K22(E))
- *Job Management Partner 1 Version 8 Job Management Partner 1/Automatic Job Management System 2 Setup Guide* (3020-3-K23(E))
- *Job Management Partner 1 Version 8 Job Management Partner 1/Automatic Job Management System 2 Operator's Guide* (3020-3-K24(E))
- *Job Management Partner 1 Version 8 Job Management Partner 1/Automatic Job Management System 2 Linkage Guide* (3020-3-K27(E))

About JP1:

- *JP1 Version 11 JP1/Performance Management/SNMP System Observer Description, Operator's Guide and Reference (3021-3-A77(E))*
- *Job Management Partner 1 Version 9 Job Management Partner 1/Software Distribution Administrator's Guide Volume 1 (3020-3-S81(E))*
- *Job Management Partner 1 Version 7i Job Management Partner 1/Performance Management/SNMP System Observer Description, Operator's Guide and Reference (3020-3-F69(E))*
- *Job Management Partner 1 Version 6 Job Management Partner 1/Server System Observer Description, User's Guide and Reference (3000-3-749(E))*

F.2 Conventions: Abbreviations for product names

This manual uses the following abbreviations for product names:

Abbreviation		Full name or meaning
IPF		Itanium(R) Processor Family
JP1/AJS2 [#]	JP1/AJS2 - Agent	Job Management Partner 1/Automatic Job Management System 2 - Agent
	JP1/AJS2 - Client Toolkit	Job Management Partner 1/Automatic Job Management System 2 - Client Toolkit
	JP1/AJS2 - Manager	Job Management Partner 1/Automatic Job Management System 2 - Manager
	JP1/AJS2 - View	Job Management Partner 1/Automatic Job Management System 2 - View
JP1/AJS3 [#]	JP1/AJS3 - Agent	JP1/Automatic Job Management System 3 - Agent
	JP1/AJS3 - Manager	JP1/Automatic Job Management System 3 - Manager
	JP1/AJS3 - View	JP1/Automatic Job Management System 3 - View
JP1/Cm2/SSO		JP1/Consolidated Management 2/SNMP System Observer
JP1/IM		JP1/Integrated Management - Manager
		JP1/Integrated Management - View
JP1/NetBatch		JP1/Network Batch Queuing System
JP1/NETM/DM	JP1/NETM/DM Client	Job Management Partner 1/NETM/DM Client
		Job Management Partner 1/NETM/DM Client - Base
	JP1/NETM/DM Client - Delivery Feature	
	JP1/NETM/DM Client - Operation Log Feature	
	JP1/NETM/DM Client - Remote Control Feature	
JP1/NETM/DM Manager		

#

In this manual, JP1/AJS is sometimes used generically, referring to JP1/AJS3 and JP1/AJS2.

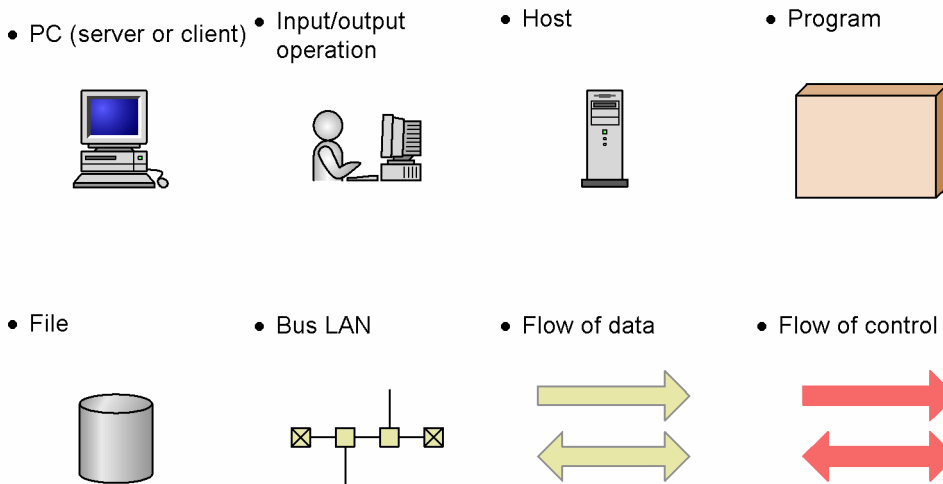
F.3 Online manuals

JP1/Script provides online help.

This online help provides the same information as provided in this manual.

F.4 Conventions: Diagrams

This manual uses the following conventions in diagrams:



F.5 Conventions: Fonts

The following table explains the text formatting conventions used in this manual:

Text formatting	Convention
Bold	<p>Bold characters indicate text in a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example:</p> <ul style="list-style-type: none"> • From the File menu, choose Open. • Click the Cancel button. • In the Enter name entry box, type your name.
<i>Italic</i>	<p>Italic characters indicate a placeholder for some actual text to be provided by the user or system. For example:</p> <ul style="list-style-type: none"> • Write the command as follows: <code>copy source-file target-file</code> • The following message appears: A file was not found. (file = <i>file-name</i>) <p>Italic characters are also used for emphasis. For example:</p> <ul style="list-style-type: none"> • Do <i>not</i> delete the configuration file.
Monospace	<p>Monospace characters indicate text that the user enters without change, or text (such as messages) output by the system. For example:</p>

Text formatting	Convention
Monospace	<ul style="list-style-type: none"> • At the prompt, enter <code>dir</code>. • Use the <code>send</code> command to send mail. • The following message is displayed: <pre>The password is incorrect.</pre>

F.6 Conventions: Symbols

The following table explains the symbols used for statements and commands in scripts. It also contains command line format explanations.

Symbol	Convention
[]	In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: [A] means that you can specify A or nothing. [B C] means that you can specify B, or C, or nothing.
...	In coding, an ellipsis (...) indicates that one or more lines of coding have been omitted. In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: A, B, B, ... means that, after you specify A, B, you can specify B as many times as necessary.
	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: A B C means A, or B, or C.
{ }	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected. For example: {A B C} means only one of A, or B, or C.

F.7 About registry key names in 64-bit versions of Windows

All registry key names used in this manual are those used in 32-bit versions of Windows.

For 64-bit versions of Windows, replace `HKEY_LOCAL_MACHINE\SOFTWARE\Hitachi\JP1/Script` with `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Hitachi\JP1/Script`.

You do not need to replace `HKEY_CURRENT_USER\SOFTWARE\HITACHI\JP1/Script`.

F.8 About the ProgramData folder

In this manual, the ProgramData folder names are those used when the default value `system-drive\ProgramData` is set for the environment variable `ALLUSERSPROFILE`.

If a value other than the default is set for the environment variable `ALLUSERSPROFILE`, read `system-drive\ProgramData` as the set value instead.

F.9 Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is 1,024² bytes
- 1 GB (gigabyte) is 1,024³ bytes.
- 1 TB (terabyte) is 1,024⁴ bytes.

G. Glossary

analysis trace file

A file that contains the results of syntactical analysis of a script.

automatic start information file

A file that contains information for the Script launcher to execute scripts automatically.

batch job

A job that uses batch processing.

batch processing

A set of processing executed as one job. Batch processing is suitable for jobs that do not require interactive operations and for jobs that process mass data.

control

A generic name for buttons, list boxes, and other items arranged on a menu form.

Easy Input

One of the component programs of JP1/Script. The user enters commands interactively, and the input commands can be output to Editor or the clipboard.

edit mode

One of the modes of Editor, used for creating a script file.

Edit mode is the default at Editor startup. To switch to monitoring mode, choose **Monitoring, Execute Monitoring**. To switch back to edit mode, choose **Monitoring, Cancel Monitoring**.

Editor

An editor with the JP1/Script attribute. Editor can be used in combination with the Easy Input facility for efficient script creation. It also provides a monitoring function that simplifies debugging by allowing the user to track script operation.

execution environment file

A file that contains the environment settings for running script files.

Execution Environment File Converter

A tool used to export execution environment files from JP1/Script for Windows to JP1/Script for UNIX.

execution trace file

A file that stores script execution results.

field name

A name set for a control. Field names are written in a menu information file, and do not appear in the created menu form.

global variable file

A file that contains the global variables set at script execution.

index number

A number from 1 for representing an element in an array variable (or a row element and column element of a two-dimensional array).

intrinsic constant

A value predefined in JP1/Script. Intrinsic constants can be used in command arguments.

JP1/Automatic Job Management System 2 (JP1/AJS2)

One of the JP1 family of products. With the product reorganization in JP1 Version 6, JP1/AJS2 now incorporates the previous JP1/AJS and JP1/NetBatch programs. JP1/Script can be linked with JP1/AJS to perform distributed processing across multiple PCs.

JP1/Automatic Job Management System 3 (JP1/AJS3)

A successor product of JP1/AJS2

JP1/Automatic Job Scheduler (JP1/AJS)

One of the JP1 family of products. JP1/AJS provides unattended operation facilities to automate and facilitate the execution of complex tasks on a PC. JP1/Script can be linked with JP1/AJS2 to perform distributed processing across multiple PCs.

JP1/Base

One of the JP1 family of products. With the product reorganization in JP1 Version 6, the event service that was formerly part of JP1/IM was moved to JP1/Base. In JP1/Script, command execution results and other information can be notified as events to JP1/Base. References to JP1/IM in the documentation also refer to JP1/Base.

JP1/Integrated Manager (JP1/IM)

One of the JP1 family of products, enabling integrated management of a distributed system. Command execution results and other information sent from JP1/Script can be displayed on the JP1/IM event console. With the product reorganization in JP1 Version 6, the JP1/IM event service was moved to JP1/Base. References to JP1/IM in the documentation also refer to JP1/Base.

JP1/Network Batch Queuing System (JP1/NetBatch)

One of the JP1 family of products, designed to run batch jobs efficiently on a PC. JP1/Script can be linked with JP1/NetBatch to perform distributed processing across multiple PCs.

JP1/Script service

One of the component programs of JP1/Script. This program controls start and termination of script files registered in the automatic start information in the service space and `NetExec` command requests in the service space.

large file

A file whose size is 2,147,483,648 bytes or greater.

location variable

A variable that contains the script file name or a parameter specified when starting a script from a command line. Location variables are in the form %0, %1, %2, . . . ,

%n, where n is a positive integer. The script file name and parameters specified in the command line are set in order in these variables, starting from %0.

For example, if the command line is entered as follows:

```
C:\TEST.SPT ABC 123
```

The following values will be set in the location variables %0, %1, and %2:

```
%0=C:\TEST.SPT
```

```
%1=ABC
```

```
%2=123
```

In this way, you can set values for the location variables when starting a script.

logon space

A script process session started by the user using Script Launcher, Script Manager, or Explorer when the user is logged on. Interactive operations using the GUI are possible.

If the user logs off during script execution, the execution terminates. Scripts that you want to execute irrespective of whether the user is logged on must be executed in the service space.

Manager

One of the component programs of JP1/Script. Manager is used for tasks such as creating and editing scripts, and setting the operating environment.

Menu Editor

An editor used to create and edit menu forms for a script file created with JP1/Script. Menu Editor also allows you to set control properties and to display menu forms in test view.

menu form

Behaves in the same way as a window. Menu forms can be created for specific purposes using Menu Editor. Buttons, list boxes, and other controls can be placed on a menu form.

menu information file

A file that contains property definitions of the forms and controls that make up a menu form. A menu information file is created automatically in text format when you edit a menu form using Menu Editor.

monitoring information file

A file that contains information used for monitoring.

monitoring mode

One of the modes of Editor, used for tracking the execution of a script file.

Edit mode is the default at Editor startup. To switch to monitoring mode, choose **Monitoring, Execute Monitoring**. To switch back to edit mode, choose **Monitoring, Cancel Monitoring**.

primary session

A session used by a physical console connected to a computer.

procedure

A set of commands processed at run time as a single unit. In JP1/Script, you can use the `Function` statement or `Sub` statement to define a procedure. The term *sub-routine* is synonymous with *procedure*.

procedure level

Indicates that a statement is written in a `Function` procedure or `Sub` procedure. A declaration, such as an assignment statement, must precede code that specifies actual operation.

Process Viewer

One of the component programs of JP1/Script. Process Viewer is used to monitor and control active script processes. Script processes running on a remote computer can also be listed in Process Viewer.

program group

A folder in which to register the shortcuts of the programs displayed in the Windows **Start** menu. Folders for specific users and a folder shared by all users are provided.

queue

A location for jobs that are awaiting execution. Submitted jobs are temporarily stored in a queue, and are then retrieved and executed in turn.

reserved keyword rule file

A file that stores the rules relating to the keywords required for analyzing and executing script files.

script

A set of coded statements, containing supplied commands and instructions.

Script Engine

One of the component programs of JP1/Script. Script Engine checks the command syntax (lexical and syntactical analysis) for created script files.

Script Execution

One of the component programs of JP1/Script. This program executes script files and processes JP1/Script commands.

Script Execution

One of the component programs of JP1/Script. This program executes created script files and processes JP1/Script commands.

script file

A file that stores a created script.

Script Launcher

One of the component programs of JP1/Script. Script Launcher controls start and termination of script files according to the predefined automatic start information.

Script Launcher service

One of the component programs of JP1/Script. This program controls `NetExec` command requests in the logon space.

script level

Indicates that a statement is written outside a procedure. In contrast with the procedure level, statements written outside a procedure are referenced as script level statements.

server environment file

A file that contains the server environment settings.

server trace file

A file that stores the results of command execution by the server when called from a command written in a script running on a client.

service information

An object that defines information necessary for service operations.

To use JP1/Script service operation commands, use the `ServiceSetValue` command to specify the necessary service information in advance.

service space

A script process session started from a JP1/Script service. By starting the script process session from the JP1/Script service, scripts can be executed irrespective of whether the user is logged on. However, interactive operation using, for example, the GUI, cannot be performed. If you need interactive operation that uses the GUI, use the logon space. Script processes started from JP1/AJS are also executed in the service space.

subroutine

Synonymous with *procedure*.

trace management file

A file that manages trace files.

Trace Viewer

One of the component programs of JP1/Script. By activating Trace Viewer, you can view the traces produced at script file execution and check the execution status.

user trace file (trace file)

A file that stores the traces produced at execution of the commands in a script.

wildcard

An asterisk (*) that represents any string, or a question mark (?) that represents any one character. In JP1/Script, a wildcard can be written anywhere in a folder name or file name. Example:

Wildcards matching `ABCDE.TXT` in the folder `_TEMP_`:

```
_TEMP_*
```

```
_TEMP_*.*
```

TEMP+"ABC*"

TEMP+"*.TXT"

TEMP+"*E*"

work file

A file used internally by JP1/Script.

Index

Symbols

- `_ALLRIGHT_` 356
- `_ARGV_` 357
- `_ARGV_CNT_` 357
- `_BIN_` 356
- `_COMP_` 356
- `_COPY_CNT_` 357
- `_COPY_RTN_` 357
- `_COPY_SKIP_CNT_` 357
- `_COPY_SKIP2_CNT_` 357
- `_DLL_RTN_` 357
- `_DOMAIN_` 356
- `_ERR_ACCESS_` 358
- `_ERR_EOF_` 357
- `_ERR_EXCLUSIVE_` 358
- `_ERR_FILE_` 358
- `_ERR_FILE_POSITION_` 358
- `_ERR_FILE_SIZE_` 358
- `_ERR_NOT_LARGE_FILE_` 358
- `_ERR_PATH_` 358
- `_ERR_PROTECT_` 358
- `_ERR_READY_` 358
- `_ERR_SERVICE_NOT_BEGIN_`
 - meaning 358
 - when to set 43
- `_ERR_SVR_CONNECT_` 358
- `_ERR_SVR_NODATA_` 358
- `_ERR_SVR_RECEIVEDATA_` 358
- `_ERR_SVR_TIMEOUT_` 358
- `_ERR_TIMEOUT_` 358
- `_EXEC_ID_` 357
- `_EXEC_RTN_` 357
- `_FORM_FIELD_NAME_` 357
- `_FORM_MODIFY_KEY_` 357
- `_FORM_TERM_CMDNO_` 357
- `_FORM_TERM_KEY_` 357
- `_JOB_RTN_` 357
- `_MSG_RTN_` 357
- `_NL_` 357
- `_NO_ERR_` 357
- `_OS_` 356
- `_OS_PLATFORM_` 356
- `_OS_REVISION_` 356
- `_OS_VERSION_` 356
- `_PROC_ID_` 357
- `_RTN_` 357
- `_RTNxx_` 357
- `_SCF_` 356
- `_SCF_EXT_` 356
- `_SCF_FIL_` 356
- `_SDF_EXT_` 356
- `_SNF_EXT_` 356
- `_SVC_RTN_` 357
- `_SVF_EXT_` 356
- `_SYS_` 356
- `_TAB_` 357
- `_TEMP_` 356
- `_USER_` 356
- `_WIN_` 356
- `_WINSYS_` 357
- `/= operator (division)` 503
- `/NOEVLOG (or /noevlog)` 375
- `/ operator (division)` 503
- `/SPALV(n) (or /spalv(n))` 375
- `/SPXLV(n) (or /spxlv(n))` 375
- `.CONF` 28
- `.LOG` 28, 29
- `.SPA` 27
- `.SPB` 28
- `.SPD` 28
- `.SPG` 28
- `.SPH` 27
- `.SPN` 28
- `.SPR` 28
- `.SPS` 27
- `.SPT` 27
- `.SPU` 28
- `.SPV` 27
- `.SPX` 27
- `.SPY` 28
- `.TMP` 28
- `.TXT` 28
- `^= operator (power)` 506
- `^ operator (power)` 505
- `'` 536
- `*= operator (multiplication)` 502
- `* operator (multiplication)` 501
- `\= operator (integer division)` 505
- `\ operator (integer division)` 504

- &= operator 422
- & operator 421
- += operator (addition) 498
- + operator 420
- + operator (addition) 497
- = operator (subtraction) 499
- operator (subtraction/negation) 499

A

- abbreviations for products 623
- accounts for communication with other computers 63
- active script processes, listing 218
- Add Files dialog box 231
- adding
 - dates 436
 - script 97
 - times 440
 - variable (to Watch window) 147
- AddStr 423
- Add Variable dialog box 272
- adjusting, controls to same size 203
- Alert 537
- aligning, controls 193
- allocating
 - space for array variable 402
 - space for variable 402
- AlreadyRun 367
- analysis trace file
 - file type 27
 - output format of 601
- analyzing, full path 474
- And operator (logical AND) 508
- array variable 358
 - coding convention of 358
 - data structure example for 362
 - declaring 402
- assigning
 - difference between variable and expression to variable 499
 - division remainder of variable and expression to variable 501
 - product of variable and expression to variable 502
 - quotient of variable and expression to variable 503, 505
 - sum of variable and expression to variable 498
 - variable raised to power of expression to variable 506
- automatic start information file 27

- automatic startup 73
 - canceling for script file 534
 - registering for script file 533

B

- backup 350
 - for operating environment information 351
 - of files used by JP1/Script 350
 - target files for 350
- basic commands 395
 - list of 396
- Beep 538
- bitmap
 - drawing 549
 - erasing 550
- BitmapHide 550
- BitmapShow 549
- breakpoint
 - canceling (in monitoring mode) 143
 - setting (in monitoring mode) 143
- Browse Button Properties (Background) dialog box 298
- Browse Button Properties (Common Items) dialog box 296
- Browse Button Properties (General Items) dialog box 295
- Browse Button Properties (Key) dialog box 299
- Browse Button Properties (Style) dialog box 300
- Button Properties (Background) dialog box 292
- Button Properties (Common Items) dialog box 290
- Button Properties (General Items) dialog box 289
- Button Properties (Key) dialog box 293
- Button Properties (Style) dialog box 294

C

- CalcDate 436
- CalcTime 440
- calculating
 - difference between dates 439
 - difference between times 442
 - length of string 414
- CallDll 568
- calling
 - DLL file 568
 - executable file 598
 - executable file on local PC 520, 522
 - executable file on remote PC 522
 - script file with Exec command 373

- script file with multiple parameters set 528
- script file with NetExec command 373
- CallSpt 528
- canceling
 - automatic startup for script file 534
 - breakpoint (in monitoring mode) 143
 - comment line 124
 - lock error function 582
 - monitoring 135
 - trial-open function 584
- CancelStartUp 534
- CatFiles 469
- centering, controls (on menu form) 201
- Change as Batch dialog box 331
- Change Folder dialog box 248
- changing
 - control attributes (batch operation) 190
 - service setting 563
- changing system date and time 76
- CheckDirName 517
- CheckDriveType 518
- checking
 - drive type 518
 - errors in SPTO control 596
 - file attribute 515
 - folder attribute 515
 - for shortcuts in program group 553
 - leap year 443
 - script syntax 94, 134
 - whether file exists 514
 - whether folder exists 513
 - whether folder is empty 512
 - whether folder is writeable 514
 - whether folder name ends with backslash 517
 - whether registry subkey exists 552
 - whether registry subkey is empty 551
 - whether service exists 552
 - whether string is lowercase 428
 - whether string is one-byte characters 430
 - whether string is two-byte characters 431
 - whether string is uppercase 429
 - whether value is numeric 512
 - whether variable is defined 511
 - whether variable is Empty value 511
- choosing, editor 86
- clearing
 - trace file 170
- user error icon 537
- closing
 - script control manager 587
 - text file 450
- coding convention
 - numeric 363
 - of array variable 358
 - of command line 376
 - of JP1/Script 352
 - script 364
 - string 363
- coding example of script control interface 591
- Combo Box Properties (Common Items) dialog box 319
- Combo Box Properties (General Items) dialog box 318
- Combo Box Properties (Key) dialog box 321
- Combo Box Properties (Style) dialog box 322
- command line
 - coding convention of 376
 - format of 373
 - parameter of 374
 - rules for writing 373
 - written in executable form (SPTXE.EXE) in user program 373
 - written in Set Execution Environment dialog box 373
- Command Line Parameter Settings dialog box 271
- command properties, defining 213
- Command Properties dialog box 324
- commands
 - entering 156
 - for automatic startup 533
 - for calling external program (special commands) 568
 - for calling external programs (basic commands) 520
 - for displaying graphics 549
 - for managing files and folders 445
 - for manipulating strings 412
 - for manipulating variables 402
 - for menu display 495
 - for message output 483
 - for performing calculations 497
 - for performing evaluations 551
 - for performing evaluations (basic commands) 511
 - for performing shortcuts 571
 - for process monitoring 576
 - for registry operations 544
 - for service operations 555
 - for working with dates 432
 - other (basic command) 537

- other (special command) 580
- comment, writing in program 536
- comment line
 - canceling 124
 - setting 124
- comparing
 - dates 438
 - files for more recent version or date 516
 - times 441
 - two expressions 507
- comparison operators (=, <>, <, <=, >, >=) 507
- CompDate 438
- CompTime 441
- concatenating
 - strings 422
 - strings with delimiters inserted 423
- Confirmation (delete trace file) dialog box 274
- Confirm File Deletion dialog box 232
 - components 233
 - operations 233
- Confirm File Overwrite dialog box 231
 - components 232
 - operations 232
- constants 363
- control attributes, changing (batch operation) 190
- control command, sending to service 566
- controls
 - adjusting to same size 203
 - aligning 193
 - centering (on menu form) 201
 - moving to background 212
 - moving to foreground 211
 - resizing to text size 206
 - spacing (equal spacing) 199
- conventions
 - abbreviations for products 623
 - diagrams 624
 - fonts 624
 - KB, MB, GB, and TB 626
 - symbols 625
- converter, files handled by 28
- converting
 - execution environment file 220
 - number to string 426
 - one-byte alphabetic characters to lowercase 415
 - one-byte alphabetic characters to uppercase 415
 - value to formatted string 427

- Copy 479
- Copy Files dialog box 230
- copying
 - file 479
 - menu form 188
 - script 96
- counting
 - elements in array variable 410
 - number of separate strings 424
- Create New Script File dialog box 228
- creating
 - folder 455
 - full path 474
 - menu form 187
 - menu form (Editor operations) 131
 - menu form (Manager operations) 112
 - program group 571
 - script 87
 - script with Easy Input 89
 - shortcut 572
 - temporary file 460

D

- Date 432
- dates
 - adding 436
 - calculating difference between 439
 - comparing 438
 - subtracting 436
- Day 433
- declaring
 - array variable 402
 - variable 402
- defining, command properties 213
- DeleteDir 456
- DeleteFile 457
- DeleteGroup 571
- DeleteGV 409, 598
- DeleteShortcut 574
- deleting
 - file 457
 - folder 456
 - global variable 409
 - global variable (DeleteGV method) 598
 - menu form 189
 - program group 571
 - registry subkey 547

- script 99
- service 559
- shortcut 574
- trace file 168
- diagram conventions 624
- differences between Exec, NetExec, and CallSpt commands 401
- Dim 402
- Dim (array) 402
- directory, list of 343
- disk volume label
 - getting 478
 - setting 477
- displaying
 - grid (on menu form) 207
 - menu form (in test view) 214
 - message 483
 - message in dialog box 488
 - text boxes 483
 - trace file contents 166
 - user-defined menu form 495
 - user error icon 537
- DLL file, calling 568
- downgrading 62
- drawing, bitmap 549

E

- Easy Input 25
 - creating script with 89
 - operations 155
 - starting 155
 - using 127
- Edit Control Properties (Common Items) dialog box 303
- Edit Control Properties (General Items) dialog box 301
- Edit Control Properties (Key) dialog box 305
- Edit Control Properties (Style) dialog box 306
- editing, script 94
- editor
 - choosing 86
- Editor 25
 - files handled by 28
 - operations 115
 - starting 72, 114
 - stopping 74, 114
- Editor operating environment, setting 132
- EnableErrorMessage property 596
- entering

- commands 156
- initialization file value 446
- registry value 545
- statements 156
- EntryStartUp 533
- environment setup
 - in a cluster system environment 64
 - in terminal service environment 68
- environment variable
 - getting 405
 - setting 403
- erasing, bitmap 550
- Error 367
- error, checking for (in SPTO control) 596
- error code
 - set by JP1/Script 613
 - set by OS 613
- error detail code, returning (for method) 595
- error message
 - getting (SPTHGetErrorMessage) 588
- error message, getting 540
- error retry function
 - canceling 582
- Error window 121
- event log 343
 - of JP1/Script 368
- ExAbortError 367
- Exec 520, 598
- Exec command, calling script file 373
- EXECID property 596
- EXECRTN property 595
- executable file
 - calling 598
 - getting execution status of 527
 - returning exit code of 595
 - returning identifier of 596
 - terminating (forcible termination) 526
 - waiting for completion of 526
- executable folder path
 - getting 476
 - setting 475
- executing
 - Execution Environment File Converter 220
 - script (Editor operations) 143
 - script (Manager operations) 109
- Execution dialog box 234
 - components 234

- operations 234
- processing 234
- execution environment file 27
- Execution Environment File Converter 25
 - executing 220
 - operations 220
 - overview of 220
- execution environment syntax file 28
- execution environment syntax file (.SPU)
 - details of 223
 - overview of 223
- execution trace file
 - file type 27
 - output format of 603
- Exit 539
- exit code
 - of JP1/Script 367
 - returning (for executable file) 595
- ExitWindows 580

F

- file
 - copying 479
 - deleting 457
 - getting version information for 466
 - joining 469
 - renaming 458
 - searching for text 151
 - sizes of 29
 - splitting 468
- file date and time
 - getting 464
 - setting 463
- File Properties dialog box 249
- file types 27
 - files handled by converter 28
 - files handled by Editor 28
 - files handled by Menu Editor 28
 - files handled by Trace Viewer 27
 - other 28
- finding
 - difference between two numbers 499
 - division remainder of two numbers 500
 - logical AND of two expressions 508
 - logical NOT of expression 510
 - logical OR of two expressions 509
 - position of string in array variable 413

- position of substring in string 412
- power of two numbers 505
- product of two numbers 501
- quotient of two numbers 503, 504
- sum of two expressions 497
- folder
 - creating 455
 - deleting 456
- font conventions 624
- Format 427
- full path
 - analyzing 474
 - creating 474
- Function Key Properties (Common Items) dialog box 310
- Function Key Properties (Style) dialog box 312

G

- GB meaning 626
- GetArrayCount 410
- GetDateCount 439
- GetDiskFreeSpace 478
- GetEnv 405
- GetEnvironment 405
- GetErrorMessage 540
- GetExecStatus 527
- GetFileAttr 462
- GetFileAttribute 462
- GetFileSize 465
- GetFileTime 464
- GetGV 407, 597
- GetPath 476
- GetProcessCount 576
- GetProcessInfo 577
- GetServiceName 567
- GetTextPosition 454
- GetTimeCount 442
- getting
 - disk free space 478
 - disk volume label 478
 - environment variable 405
 - error message 540
 - error message (SPTHGetErrorMessage) 588
 - execution status of executable file 527
 - file attributes 462
 - file date and time 464
 - file size 465

- folder attributes 462
- global variable 407
- global variable (GetGV method) 597
- information set for active service 564
- number of process activations 576
- path to executable folder 476
- process information 577
- service name from service display name 567
- service status 565
- temporary folder name 459
- values in service information 557
- version information for file 466
- GetVerInfo 466
- GetVersionInfo 466
- GetVolLabel 478
- GetVolumeLabel 478
- global variable
 - deleting 409
 - deleting (DeleteGV method) 598
 - getting 407
 - getting (GetGV method) 597
 - setting 406
 - setting (SetGV method) 597
- global variable file 28
- GrammarError 367
- grid, displaying on menu form 207

H

- halting
 - script execution 537
 - service 562
- hiding, trace file 165
- Hour 435

I

- identifier, returning (for executable file) 596
- IMEventMessage 492
- InArray 413
- indent size, setting 158
- IniRead 445
- initialization file value
 - entering 446
 - reading 445
- IniWrite 446
- InputBox 483
- installing, JP1/Script 55

- InStr 412
- IP address, changing 69
- IsDef 511
- IsDefine 511
- IsEmpty 511
- IsEmptyDir 512
- IsEmptyGroup 553
- IsEmptyReg 551
- IsExistDir 513
- IsExistFile 514
- IsExistRegKey 552
- IsExistService 552
- IsFileAttr 515
- IsFileAttribute 515
- IsLeapYear 443
- IsLower 428
- IsMultiChar 431
- IsNew 516
- IsNumeric 512
- IsSingleChar 430
- issuing, event to JP1/IM or JP1/Base 492
- IsUpper 429
- IsWriteableDir 514

J

- joining, split files 469
- JP1/Script
 - coding convention of 352
 - component programs of 25
 - error code set by 613
 - event log of 368
 - exit code of 367
 - features of 23
 - files used in 27
 - forcibly terminating, from JP1/AJS 74
 - installing 55
 - operating 80
 - operating procedure for 39
 - overall organization of 24
 - overview of 22
 - preparation for using 54
 - reinstalling 60
 - starting 72
 - stopping 72, 74
 - system configuration of 35
 - uninstalling 55, 59
 - upgrade installation 61

- JP1/Script dialog boxes 227
- JP1/Script service 25
- JP1/Script system configuration
 - linked with JP1/AJS 35
 - linked with JP1/Base 35
 - linked with JP1/IM 36
 - using JP1/Script in cluster environment 37
- JP1/Script version for script creation, specifying 157

K

- KB meaning 626
- key operations
 - in Script Editor window 123
 - in Script Manager window 86
 - in Script Menu Editor window 185
 - in Script Trace Files Display window 177
 - in Script Trace Viewer window 163
- keyword, restricted 354

L

- large file 31
- LCase 415
- Left 416
- Len 414
- line length, setting 158
- Line Properties (General Items) dialog box 308
- Link Editor dialog box 252
- List Control Properties (Common Items) dialog box 314
- List Control Properties (General Items) dialog box 313
- List Control Properties (Key) dialog box 316
- List Control Properties (Style) dialog box 317
- listing, active script processes 218
- lock error retry function 49
 - setting 581
- log file, list of 343
- logging off, Windows 580
- log information 342
 - types of 342
- LTrim 418

M

- maintenance log file 28
- MakeDir 455
- MakeGroup 571
- MakePath 474
- MakeShortcut 572

- Manager 25
 - files handled by 27
 - operations 81
 - starting 72
 - stopping 74
- MB meaning 626
- Menu 495
- Menu Editor 25
 - files handled by 28
 - operations 178
 - starting 73
 - stopping 74
- menu form 187
 - copying 188
 - creating 187
 - creating (Editor operations) 131
 - creating (Manager operations) 112
 - deleting 189
 - displaying in test view 214
 - pasting 189
 - printing 215
- Menu Form Properties (Background) dialog box 279
- Menu Form Properties (General Items) dialog box 278
- Menu Form Properties (Key) dialog box 280
- Menu Form Properties (Style) dialog box 282
- Menu Form Properties (Wallpaper) dialog box 283
- menu information file 28
- menus
 - in Script Editor window 119
 - in Script Manager window 83
 - in Script Menu Editor window 182
 - in Script Trace Files Display window 176
 - in Script Trace Viewer window 162
- Message 484
- MessageBox 488
- MessageEventLog 490
- Mid 416
- Minute 435
- Mod= operator (remainder calculation) 501
- Mod operator (remainder calculation) 500
- monitoring
 - canceling 135
 - JP1/Script by activity monitoring program 70
 - starting 135
- monitoring information file 28
- Month 433
- mouse operations

- in Script Editor window 123
- in Script Manager window 85
- in Script Menu Editor window 185
- in Script Trace Files Display window 176
- in Script Trace Viewer window 163

moving

- control (to background) 212
- control (to foreground) 211
- read/write position to beginning of text file 453
- read/write position to end of text file 453

N

- naming convention, variable 353
- NetExec 522
- NetExec command, calling script file 373
- NetExec command execution control 25
- NetExec command restriction policy file 28
- NetExec error log file 343
 - output format of 608
- Not operator (logical NOT) 510
- numeric coding convention 363

O

opening

- script control manager 587
- text file 448

operating environment, setting (for monitoring mode) 146

operation conventions 364

operations

- Easy Input 155
- Editor 115
- JP1/Script 80
- Manager 81
- Menu Editor 178
- Process Viewer 216
- Script Trace Viewer window 165
- Trace Files Display 174
- Trace Viewer 160

operator precedence 364

operators

- / (division) 503
- /= (division) 503
- ^ (power) 505
- ^= (power) 506
- (subtraction/negation) 499

- * (multiplication) 501
- *= (multiplication) 502
- \ (integer division) 504
- \= (integer division) 505
- + (addition) 497
- += (addition) 498
- = (subtraction) 499
- And (logical AND) 508
- Mod (remainder calculation) 500
- Mod= (remainder calculation) 501
- Not (logical NOT) 510
- Or (logical OR) 509

- Options (Cluster Environment) dialog box 259
- Options (Colors) dialog box 265
- Options (Compatibility) dialog box 254, 267
- Options (Format) dialog box 264
- Options (JP1/IM) dialog box 257
- Options (Multi-activation) dialog box 255
- Options (Server Information) dialog box 252
- Options (Trace) dialog box 258
- Or operator (logical OR) 509
- OS, error codes set by 613
- output format
 - of analysis trace file 601
- output format
 - of execution trace file 603
 - of NetExec error log file 608
 - of Script trace file 601
 - of server trace file 606
 - of user trace file 606
- outputting
 - message to application log at Event Viewer 490
 - text to file or window 484

P

- parameter of command line 374
- parameters 374
- pasting, menu form 189
- Print Information of Menu Forms dialog box 334
- printing
 - menu form 215
 - trace file 171
- process
 - getting information of 577
 - terminating (forcible termination) 578
- process activations, getting number of 576
- Process Viewer 25

- operations 216
- starting 216
- troubleshooting 341
- Program execution information file 29
- Program execution information management file 28
- program group
 - creating 571
 - deleting 571
- program installation folder 55

R

- read/write position
 - moving to beginning of text file 453
 - moving to end of text file 453
 - returning 454
- reading
 - initialization file value 445
 - one line of data from text file 451
 - registry value 544
- recovering 350
 - files used by JP1/Script 350
 - operating environment information 351
- refresh interval for listing script processes, specifying 218
- RegDelete 546
- RegDeleteKey 547
- registering
 - script file for automatic startup 533
 - service 559
- Registering the JP1/Script service 59
- registry subkey
 - checking empty status 551
 - deleting 547
- registry value
 - deleting 546
 - entering 545
 - reading 544
- RegRead 544
- RegWrite 545
- reinstalling
 - JP1/Script 60
- Rem 536
- removing
 - leading spaces from string 418, 420
 - trailing spaces from string 419, 420
- Rename 458
- Rename dialog box 233

- components 233
- operations 233
- processing 233
- renaming
 - file 458
 - hosts 69
 - script 100
- replacing
 - string in text file 447
 - text 153
- reserved keyword rule file 28
- ResetRetryMode 582
- ResetStandardFile 473
- ResetStdFile 473
- resetting
 - standard error file 473
 - standard input file 473
 - standard output file 473
- ResetTrialOpenMode 584
- resizing, control (to text size) 206
- resuming, service 563
- returning
 - characters from inside string 416
 - characters from left side of string 416
 - characters from right side of string 417
 - current date 432
 - current read/write position 454
 - current time 432
 - day for specified date 433
 - error detail code for method 595
 - exit code of executable file 595
 - hour for specified time 435
 - identifier of executable file 596
 - minute for specified time 435
 - month for specified date 433
 - second for specified time 436
 - specified number of one-byte spaces 418
 - weekday for specified date 434
 - year for specified date 432
- Right 417
- RTN property 595
- RTrim 419

S

- sample files 611
- saving
 - script 87

- trace file (under new name) 167
- script
 - adding 97
 - checking syntax of 94
 - coding convention 364
 - copying 96
 - creating 87
 - creating with Easy Input 89
 - deleting 99
 - editing 94
 - executing (Editor operations) 143
 - executing (Manager operations) 109
 - halting execution of 537
 - renaming 100
 - rules for creating 353
 - saving 87
 - starting automatically 108
 - terminating 580
 - terminating execution of 539
- script control functions 587
- script control interface 585
 - about 586
 - coding example of 591
- script control manager
 - closing 587
 - opening 587
- Script Editor dialog boxes 263
 - Add Variable dialog box 272
 - Command Line Parameter Settings dialog box 271
 - Options (Colors) dialog box 265
 - Options (Compatibility) dialog box 267
 - Options (Format) dialog box 264
 - Search dialog box 268
 - Set dialog box 267
 - Set File Name dialog box 269
 - Set File Version dialog box 270
 - Update Value dialog box 263
- Script Editor window 116
 - key operations in 123
 - menus in 116, 119
 - mouse operations in 123
- script execution, terminating (forcible termination) 588
- script execution control 25
- script execution environment
 - setting 148
 - setting all items 101
 - setting individual items 104
- script file 27
 - canceling automatic startup for 534
 - registering automatic startup for 533
 - terminating forcibly 74
- script launcher 25
- Script launcher service 25
- Script Manager dialog boxes 228
 - Add Files dialog box 231
 - Change Folder dialog box 248
 - Confirm File Deletion dialog box 232
 - Confirm File Overwrite dialog box 231
 - Copy Files dialog box 230
 - Create New Script File dialog box 228
 - Execution dialog box 234
 - File Properties dialog box 249
 - Link Editor dialog box 252
 - Options (Cluster Environment) dialog box 259
 - Options (Compatibility) dialog box 254
 - Options (JP1/IM) dialog box 257
 - Options (Multi-activation) dialog box 255
 - Options (Server Information) dialog box 252
 - Options (Trace) dialog box 258
 - Rename dialog box 233
 - Select Folder dialog box 261
 - Set Automatic Startup dialog box 250
 - Set Command Line dialog box 241
 - Set Execution Environment (Command Line) dialog box 240
 - Set Execution Environment (Start Date) dialog box 243
 - Set Execution Environment (Start Information) dialog box 234
 - Set Execution Environment (Terminate Information) dialog box 237
 - Set Execution Environment (Terminate Time) dialog box 245
 - Set Execution Environment (Trace Information) dialog box 238
 - Set Execution Environment (Trace Output Folder) dialog box 246
 - Set Execution Environment (User Trace Information) dialog box 247
 - Set Execution Environment (Working Folder) dialog box 242
 - Update Information dialog box 254
- Script Manager window 81
 - key operations in 85
 - menus in 81, 83
 - mouse operations in 85

- Script Menu Editor dialog boxes 277
 - Browse Button Properties (Background) dialog box 298
 - Browse Button Properties (Common Items) dialog box 296
 - Browse Button Properties (General Items) dialog box 295
 - Browse Button Properties (Key) dialog box 299
 - Browse Button Properties (Style) dialog box 300
 - Button Properties (Background) dialog box 292
 - Button Properties (Common Items) dialog box 290
 - Button Properties (General Items) dialog box 289
 - Button Properties (Key) dialog box 293
 - Button Properties (Style) dialog box 294
 - Change as Batch dialog box 331
 - Combo Box Properties (Common Items) dialog box 319
 - Combo Box Properties (General Items) dialog box 318
 - Combo Box Properties (Key) dialog box 321
 - Combo Box Properties (Style) dialog box 322
 - Command Properties dialog box 324
 - Edit Control Properties (Common Items) dialog box 303
 - Edit Control Properties (General Items) dialog box 301
 - Edit Control Properties (Key) dialog box 305
 - Edit Control Properties (Style) dialog box 306
 - Function Key Properties (Common Items) dialog box 310
 - Function Key Properties (General Items) dialog box 309
 - Function Key Properties (Style) dialog box 312
 - Line Properties (General Items) dialog box 308
 - List Control Properties (Common Items) dialog box 314
 - List Control Properties (General Items) dialog box 313
 - List Control Properties (Key) dialog box 316
 - List Control Properties (Style) dialog box 317
 - Menu Form Properties (Background) dialog box 279
 - Menu Form Properties (General Items) dialog box 278
 - Menu Form Properties (Key) dialog box 280
 - Menu Form Properties (Style) dialog box 282
 - Menu Form Properties (Wallpaper) dialog box 283
 - Print Information of Menu Forms dialog box 334
 - Set Command Properties dialog box 325
 - Set Grid dialog box 332
 - Set Menu Form Name dialog box 330
 - Set Tab Order dialog box 333
 - Static Properties (Background) dialog box 287
 - Static Properties (Common Items) dialog box 285
 - Static Properties (General Items) dialog box 284
 - Static Properties (Style) dialog box 288
- Script Menu Editor window 179
 - key operations in 185
 - menus in 179, 182
 - mouse operations in 185
- script OLE control 593, 594
- script syntax, checking 134
- Script trace file, output format of 601
- Script Trace Files Display window 174
 - key operations in 176
 - menus in 176
 - mouse operations in 176
- Script Trace Viewer dialog boxes 274
 - Confirmation (clear trace file) dialog box 274
 - Confirmation (delete trace file) dialog box 274
 - Search dialog box 276
 - Select Computer dialog box 275
- Script Trace Viewer window 160
 - key operations in 163
 - menus in 160, 162
 - mouse operations in 163
 - operations 165
- Search dialog box 268, 276
- searching
 - for text (in file) 151
 - for text (in trace file) 172
- Second 436
- Select Computer dialog box 275
- Select Folder dialog box 261
- sending, control command to service 566
- SeparateStr 425
- SeparateStrCount 424
- server environment file 27
- server trace file
 - file type 28
 - output format of 606
- service
 - changing setting for 563
 - checking existence of 552
 - deleting 559
 - getting current status of 565
 - getting information set for active 564

- halting 562
- registering 559
- resuming 563
- sending control command to 566
- starting 560
- stopping 561
- ServiceChange 563
- ServiceContinue 563
- ServiceControl 566
- ServiceCreate 559
- ServiceDelete 559
- ServiceGetValue 557
- service information
 - getting values in 557
 - setting 555
- service name, getting from service display name 567
- ServicePause 562
- ServiceQuery 564
- ServiceRefer 565
- ServiceSetValue 555
- ServiceStart 560
- ServiceStop 561
- Set Automatic Startup dialog box 250
- Set Command Line dialog box 241
- Set Command Properties dialog box 325
- Set dialog box 267
- SetEnv 403
- SetEnvironment 403
- Set Execution Environment (Command Line) dialog box 240
- Set Execution Environment (Start Date) dialog box 243
- Set Execution Environment (Start Information) dialog box 234
 - components 235
 - operations 237
 - processing 237
- Set Execution Environment (Terminate Information) dialog box 237
- Set Execution Environment (Terminate Time) dialog box 245
- Set Execution Environment (Trace Information) dialog box 238
- Set Execution Environment (Trace Output Folder) dialog box 246
- Set Execution Environment (User Trace Information) dialog box 247
- Set Execution Environment (Working Folder) dialog box 242
- Set Execution Environment dialog box, command line written in 373
- SetFileAttr 461
- SetFileAttribute 461
- Set File Name dialog box 269
- SetFileTime 463
- Set File Version dialog box 270
- Set Grid dialog box 332
- SetGV 406, 597
- Set Menu Form Name dialog box 330
- SetPath 475
- SetRetryMode 581
- SetStandardFile 471
- SetStdFile 471
- Set Tab Order dialog box 333
- setting
 - breakpoint (in monitoring mode) 143
 - comment line 124
 - disk volume label 477
 - Editor operating environment 132
 - environment variable 403
 - file attributes 461
 - file date and time 463
 - folder attributes 461
 - global variable 406
 - global variable (SetGV method) 597
 - indent size 158
 - line length 158
 - lock error retry function 581
 - operating environment (for monitoring mode) 146
 - path to executable folder 475
 - script execution environment 148
 - script execution environment (all items) 101
 - script execution environment (individual items) 104
 - service information 555
 - standard error file 471
 - standard input file 471
 - standard output file 471
 - tab order 209
 - trial-open function 583
- SetTrialOpenMode 583
- SetVollabel 477
- SetVolumeLabel 477
- shortcut
 - checking existence of, in program group 553
 - creating 572
 - deleting 574

- showing, trace file 165
- shutting down, Windows 580
- Sleep 537
- sounding, beep from speakers 538
- space
 - allocating, for array variable 402
 - allocating, for variable 402
- Space 418
- spacing, controls equally 199
- special commands 541
 - list of 542
- specifying
 - JP1/Script version (for script creation) 157
 - refresh interval for listing script processes 218
- SplitFile 468
- SplitPath 474
- splitting
 - file 468
 - string into separate strings 425
- SPTHClose 587
- SPTHGetErrorMessage 588
- SPTHOpen 587
- SPTHTerminate 588
- SPTO control 595
 - checking for errors in 596
- SPTXE.EXE 373
- standard error file
 - resetting 473
 - setting 471
- standard input file
 - resetting 473
 - setting 471
- standard output file
 - resetting 473
 - setting 471
- starting
 - Easy Input 155
 - Editor 72, 114
 - JP1/Script 72
 - Manager 72
 - Menu Editor 73
 - monitoring 135
 - Process Viewer 216
 - script (automatic startup) 108
 - service 560
 - Trace Viewer 72, 164
- statements 379
- = 380
- Call 391
- Continue 393
- Do...Loop 381
 - entering 156
- Exit xx 391
- For...End For 383
- For...Next 382
- Function 388
- GoTo 392
- If...Then...Else 384
 - list of 380
- On Error 393
- Select Case 386
- Sub 389
- While...End 387
- Static Properties (Background) dialog box 287
- Static Properties (Common Items) dialog box 285
- Static Properties (General Items) dialog box 284
- Static Properties (Style) dialog box 288
- stopping
 - Editor 74, 114
 - JP1/Script 72, 74
 - Manager 74
 - Menu Editor 74
 - service 561
 - Trace Viewer 74, 164
- Str 426
- string
 - calculating length of 414
 - coding convention 363
- subkey registry, checking existence of 552
- subtracting
 - dates 436
 - times 440
- symbol conventions 625

T

- tab order, setting 209
- TB meaning 626
- TempDir 459
- TempFile 460
- Terminate 367
- TerminateProcess 578
- terminating
 - executable file (forcible termination) 526
 - process (forcible termination) 578

- script 580
- script execution 539
- script execution (forcible termination) 588
- text, replacing 153
- TextClose 450
- text file
 - closing 450
 - opening 448
 - reading one line of data from 451
 - replacing string in 447
 - writing data to 452
- TextFileReplace 447
- TextOpen 448
- TextRead 451
- TextSeek 453
- TextWrite 452
- Time 432
- TimeOut 367
- times
 - adding 440
 - calculating difference between 442
 - comparing 441
 - subtracting 440
- trace file
 - clearing 170
 - deleting 168
 - displaying contents 166
 - hiding 165
 - printing 171
 - saving under new name 167
 - searching for text 172
 - showing 165
- Trace Files Display operations 174
- Trace Files Display window, menus of 174
- trace management file 28
- Trace Viewer 25
 - files handled by 27
 - operations 160
 - starting 72, 164
 - stopping 74, 164
 - troubleshooting 340
- trial-open function 52
 - canceling 584
 - setting 583
- Trim 420
- troubleshooting 336, 338
 - command execution problems 339

- information to be acquired in the event of an error 345
- procedure for 337
- Process Viewer problems 341
- script execution problems 338
- Trace Viewer problems 340
- type of log information
 - analysis trace file 342
 - execution trace file 342
 - server trace file 343

U

- UCase 415
- uninstalling, JP1/Script 59
- Update Information dialog box 254
- Update Value dialog box 263
- upgrade installation
 - JP1/Script 61
- user error icon
 - clearing 537
 - displaying 537
- user trace file 342
 - output format of 606
- user trace file (trace file) 28
- using Easy Input 127
- using JP1/Script system configuration
 - using JP1/Script in terminal service environment 37

V

- variable
 - adding (to Watch window) 147
 - array variable 358
 - data size of 353
 - declaring 402
 - maximum number of 353
 - naming convention 353
 - reserved 355

W

- WaitForExec 526
- waiting
 - for completion of executable file 526
- Watch window 122
 - adding variable to 147
- Weekday 434
- Windows
 - logging off 580

shutting down 580

work file 28

writing

comment in program 536

data to text file 452

Y

Year 432