



**Wavelink Telnet Client  
Scripting Commands Library**

*Revised 11/10/2009*

Copyright © 2009 by Wavelink Corporation All rights reserved.

Wavelink Corporation  
6985 South Union Park Avenue, Suite 335  
Midvale, Utah 84047  
Telephone: (801) 316-9000  
Fax: (801) 316-9099  
Email: [customerservice@wavelink.com](mailto:customerservice@wavelink.com)  
Website: <http://www.wavelink.com>

Email: [sales@wavelink.com](mailto:sales@wavelink.com)

No part of this publication may be reproduced or used in any form, or by any electrical or mechanical means, without permission in writing from Wavelink Corporation. This includes electronic or mechanical means, such as photocopying, recording, or information storage and retrieval systems. The material in this manual is subject to change without notice.

The software is provided strictly on an “as is” basis. All software, including firmware, furnished to the user is on a licensed basis. Wavelink grants to the user a non-transferable and non-exclusive license to use each software or firmware program delivered hereunder (licensed program). Except as noted below, such license may not be assigned, sublicensed, or otherwise transferred by the user without prior written consent of Wavelink. No right to copy a licensed program in whole or in part is granted, except as permitted under copyright law. The user shall not modify, merge, or incorporate any form or portion of a licensed program with other program material, create a derivative work from a licensed program, or use a licensed program in a network without written permission from Wavelink . The user agrees to maintain Wavelink’s copyright notice on the licensed programs delivered hereunder, and to include the same on any authorized copies it makes, in whole or in part. The user agrees not to decompile, disassemble, decode, or reverse engineer any licensed program delivered to the user or any portion thereof.

Wavelink reserves the right to make changes to any software or product to improve reliability, function, or design.

The information in this document is bound by the terms of the end user license agreement.

# Table of Contents

|                                     |           |
|-------------------------------------|-----------|
| <b>Introduction</b> .....           | <b>7</b>  |
| About this Document .....           | 7         |
| About Telnet Client Scripting.....  | 8         |
| <br>                                |           |
| <b>Overview of Actions</b> .....    | <b>11</b> |
| <br>                                |           |
| <b>No Return Values</b> .....       | <b>12</b> |
| Blank_Line .....                    | 17        |
| Comment.....                        | 18        |
| Goto .....                          | 19        |
| Label.....                          | 20        |
| Return .....                        | 21        |
| Abort .....                         | 22        |
| Abort_All.....                      | 23        |
| Disconnect.....                     | 24        |
| Exit_Application .....              | 25        |
| If .....                            | 26        |
| If_Not.....                         | 27        |
| Else .....                          | 28        |
| End_If.....                         | 29        |
| While .....                         | 30        |
| While_Not.....                      | 32        |
| End_While.....                      | 34        |
| Continue .....                      | 35        |
| Break .....                         | 36        |
| Ask_Ok .....                        | 37        |
| Keypress_String.....                | 38        |
| Keypress_Key.....                   | 39        |
| Scan_String .....                   | 40        |
| Set_Cursor_Position .....           | 42        |
| Message.....                        | 43        |
| Message_Clear .....                 | 44        |
| Beep .....                          | 45        |
| Play_Sound.....                     | 47        |
| Wait_For_Screen_Update .....        | 48        |
| Delay .....                         | 49        |
| Logging_On .....                    | 50        |
| Logging_Off .....                   | 52        |
| Call .....                          | 53        |
| Button_Bitmap_Create_Emulation..... | 55        |
| Button_Bitmap_Create_View.....      | 57        |
| Button_Create_Emulation .....       | 59        |

|                                 |    |
|---------------------------------|----|
| Button_Create_View .....        | 61 |
| Button_Remove .....             | 63 |
| Button_Remove_All.....          | 64 |
| Reboot .....                    | 65 |
| Keypress_Capture .....          | 66 |
| Keypress_Capture_Stop .....     | 68 |
| Keypress_Capture_Stop_All ..... | 70 |
| Keyboard_Disable .....          | 71 |

## **Boolean Values..... 72**

|                                    |     |
|------------------------------------|-----|
| Boolean_Set.....                   | 77  |
| Boolean_Not .....                  | 79  |
| Boolean_Equal .....                | 85  |
| Boolean_Not_Equal.....             | 87  |
| String_Empty .....                 | 89  |
| String_Equal .....                 | 95  |
| String_Not_Equal.....              | 101 |
| Set_Field_Data_ID.....             | 103 |
| Get_Field_Symbology_Operator.....  | 107 |
| Set_Field_Append_Scan_Data.....    | 109 |
| Set_Field_Com_Data_Field .....     | 111 |
| Set_Field_Prefix_Scan_Data.....    | 113 |
| Get_Field_Append_Scan_Data .....   | 115 |
| Number_Less_Than.....              | 117 |
| Number_Less_Than_Or_Equal .....    | 119 |
| Number_Equal.....                  | 121 |
| Number_Greater_Than_Or_Equal ..... | 123 |
| Number_Greater_Than.....           | 125 |
| Number_Not_Equal .....             | 127 |
| Ask_OK_Cancel.....                 | 129 |
| Ask_Yes_No .....                   | 131 |
| Suspend.....                       | 133 |
| Web_Navigate .....                 | 137 |
| Web_Navigate_Frame .....           | 138 |
| Web_Navigate_Post_Data .....       | 140 |
| Web_Scripting .....                | 142 |
| Web_Search_Source.....             | 144 |
| Speech_From_Text_Available .....   | 146 |
| Speech_From_Text.....              | 147 |
| Speech_To_Text_Available .....     | 149 |
| Speech_To_Text_No_Wait .....       | 152 |
| Speech_To_Text_Cancel .....        | 156 |
| Speech_Setting_Available.....      | 157 |
| Speech_Change_Setting .....        | 159 |
| Speech_Get_Setting.....            | 160 |
| Speech_Get_Setting_Max.....        | 162 |

|   |     |
|---|-----|
| Speech_Find_Setting_Value .....           | 164 |
| Speech_Get_Setting_Value_Desc.....        | 166 |
| Speech_To_Text_Get_User_Name.....         | 168 |
| Speech_To_Text_Change_User_Name .....     | 169 |
| Speech_From_Text_Error_Desc .....         | 170 |
| Speech_To_Text_Error_Desc .....           | 171 |
| Speech_From_Text_Cancel .....             | 172 |
| Wait_For_Screen_Update_With_Timeout ..... | 173 |
| Keypress_Capture_Stop .....               | 174 |
| Cancel_Other_Scripts .....                | 176 |
| Printer_Data .....                        | 177 |
| Printer_Repeat .....                      | 179 |
| Printer_Cancel .....                      | 181 |

## **String Values ..... 183**

|                                    |     |
|------------------------------------|-----|
| Get_MAC_Address .....              | 186 |
| Get_IP_Address .....               | 187 |
| Get_Screen_Text .....              | 190 |
| Get_Screen_Text_Columns .....      | 194 |
| Get_Workstation_ID .....           | 196 |
| Get_Avalanche_Property_Value ..... | 197 |
| Get_Scan_Type_Name .....           | 198 |
| Web_Get_Source .....               | 200 |
| Web_Get_Current_Element .....      | 201 |
| Escape_Sequence .....              | 202 |
| String_Set .....                   | 203 |
| String_Combine .....               | 205 |
| String_Left .....                  | 207 |
| String_Right .....                 | 209 |
| String_Upper .....                 | 213 |
| String_Lower .....                 | 215 |
| String_Replace .....               | 217 |
| String_Trim_Spaces_Start .....     | 223 |
| String_Trim_Spaces_End .....       | 224 |
| Number_To_String_Octal .....       | 228 |
| Number_To_String_Decimal .....     | 230 |
| Ask_String .....                   | 236 |
| Ask_String_Password .....          | 238 |
| Ask_String_Uppercase .....         | 240 |
| Ask_String_Lowercase .....         | 242 |
| Number_To_Character .....          | 244 |
| Get_Field_Data_ID .....            | 246 |
| Get_Field_Prefix_Scan_Data .....   | 248 |

## **Integer Values ..... 250**

|                              |     |
|------------------------------|-----|
| Get_Screen_Columns           | 254 |
| Get_Screen_Rows              | 255 |
| Get_Position_Column          | 256 |
| Get_Position_Row             | 258 |
| Get_Session_Number           | 260 |
| Get_Time                     | 261 |
| Get_Field_Column             | 274 |
| Get_Num_Field_Data_IDs       | 278 |
| Get_Num_Field_Symbology_IDs  | 280 |
| Get_Field_Com_Data_Field     | 282 |
| Get_Scan_Type_Value          | 283 |
| Ask_Yes_No_Cancel            | 285 |
| Run_Application              | 287 |
| String_Length                | 289 |
| String_Find_Last             | 292 |
| Number_Divide_Remainder      | 304 |
| String_To_Number_Binary      | 306 |
| String_To_Number_Octal       | 308 |
| String_To_Number_Decimal     | 310 |
| String_To_Number_Hexadecimal | 312 |
| Character_To_Number          | 316 |
| Bitwise_And                  | 318 |
| Bitwise_Or                   | 319 |
| Bitwise_Xor                  | 320 |
| Bitwise_Not                  | 321 |

## **Symbologies and Values . . . . . 322**

## **Voice-Enabled Emulation Settings . . . . . 324**

|                              |     |
|------------------------------|-----|
| tts_language                 | 327 |
| tts_language_short           | 329 |
| tts_language_long            | 330 |
| tts_voice                    | 331 |
| tts_frequency                | 332 |
| tts_volume                   | 333 |
| tts_rate                     | 335 |
| tts_readmode                 | 336 |
| tts_waitfactor               | 338 |
| tts_calibrate                | 340 |
| tts_external_speaker_setting | 341 |
| stt_domain                   | 343 |
| stt_language                 | 344 |
| stt_language_short           | 345 |
| stt_language_long            | 346 |
| stt_frequency                | 347 |

---

|                         |     |
|-------------------------|-----|
| stt_size                | 348 |
| stt_timeout             | 349 |
| stt_idle_timeout        | 350 |
| stt_silence             | 352 |
| stt_fx_silence          | 353 |
| stt_expanded            | 354 |
| stt_confidence          | 356 |
| stt_fx_detect_start     | 358 |
| stt_threshold           | 360 |
| stt_fx_threshold        | 362 |
| stt_save_threshold      | 364 |
| stt_fx_min_duration     | 365 |
| stt_fx_sensitivity      | 367 |
| stt_volume              | 369 |
| stt_calibrate           | 370 |
| stt_grammar_optimize    | 371 |
| stt_grammar_phonetic    | 373 |
| stt_grammar_nonterminal | 375 |
| stt_context_optimize    | 377 |
| stt_processing          | 379 |
| stt_save_session_delay  | 381 |
| stt_reset_session_delay | 383 |
| stt_special_sounds      | 385 |
| stt_fx_microphone       | 387 |
| stt_priority            | 389 |



# Introduction

The Script Editor Actions Library is a complete list of actions and settings for script development in the Telnet Client Script Editor. This document provides usage information and example code.

This introduction presents the following information:

- About this Document
- About Telnet Client Scripting

## About this Document

This section describes the assumptions and conventions of this document.

### Document Assumptions

This document makes the following assumptions:

- Wavelink Telnet Client is installed on your computer
- You are familiar with the Telnet Client Script Editor
- You have knowledge or rudimentary experience with programming/scripting languages

### Document Conventions

This section describes how different information is formatted in this document.

#### Parameters

Parameters are represented in italic Courier New font. The descriptions of the parameters, when necessary, are listed at the right.

Example:

*Return Value*

The value that the Telnet application returns to the system as it exits.

### Sample Code

Sample code is displayed in Courier New font.

Example:

```
Script(Goto_Test)
  Activate(From_Menu)
  Goto: end_script
  Message("Skipped...", 0)
  Label: end_script
  Message("The End.", 0)
  Return
```

---

**NOTE** If you copy and paste the sample code, you may need to edit out returns.

---

## About Telnet Client Scripting

The Script Editor is a component of the Wavelink Telnet Client. The Script Editor provides the ability to create and execute scripts that automate processes on the Telnet Client.

The Script Editor is also the basis for using Voice-Enabled Emulation (or Speakeasy) with your Telnet Client installation.

This section includes the following information:

- Overview of the Scripting Process
- Debugging Scripts
- Additional Information

### Overview of the Scripting Process

The following steps outline the process of creating scripts using the Script Editor:

- 1 Launch the Script Editor.** You can launch the Script Editor from the Telnet Client or the Avalanche Console.

**2 Create scripts using the Script Editor.** Use the Script Editor to manually build the script code.

-Or-

**Create scripts using the Script Capture option.** Capture the actions you want to include in your script to build the script code.

-Or-

**Create scripts from text.** Import a text file or create a text-based script using the Text Editor.

**3 Configure an execution method for the script.** Select from the available options the method you want to use to execute your script.

**4 Execute the script from the Telnet Client.** Using the activation method you selected for the script, you can activate and execute your script.

## Debugging Scripts

You can debug a Telnet Client script from the Text Editor. For details about using the Text Editor and debugging scripts, refer to the *Wavelink Telnet Client Scripting Reference Guide*.

### To debug the script:

**1** Build your script in the Text Editor. When your script is complete, click **Build** to verify the script text.

The Text Editor displays build information in the bottom section of the window.

**2** If the build information includes an error message, single-click the message to display the *Script Editor Error Help* dialog box.

The *Script Editor Error Help* dialog box displays the error number, a description of the error, and explains how to fix the error.

## Additional Information

For more information about the Wavelink Script Editor, see the following documentation:

- *Wavelink Telnet Client User Guide*

- *Wavelink Telnet Client Scripting Reference Guide*
- *Wavelink Telnet Client Voice-Enabled Emulation Reference Guide*

## Overview of Actions

The following tables display an overview of the actions in the Telnet Client Script Editor. The actions have been divided into sections according to the type of value that is returned after each action. The following is a list of the values:

- No Return Values
- Boolean Values
- String Values
- Integer Values

## No Return Values

This section contains a list of scripting actions that return no value. The following action categories are described in this section:

### Blank Line and Comment Actions

| Action                     | Description   |
|----------------------------|---|
| <a href="#">Blank_Line</a> | Proceeds to the next instruction without taking any action. |
| <a href="#">Comment</a>    | Proceeds to the next instruction without taking any action. |

### Goto Support Actions

| Action                | Description                     |
|-----------------------|---------------------------------|
| <a href="#">Goto</a>  | Jumps to the supplied label.    |
| <a href="#">Label</a> | Label to which a Goto can jump. |

### Macro Existing

| Action                           | Description  |
|----------------------------------|--|
| <a href="#">Return</a>           | Exits the script normally.                                     |
| <a href="#">Abort</a>            | Exits the script immediately.                                  |
| <a href="#">Abort_All</a>        | Exits all scripts for the session.                             |
| <a href="#">Disconnect</a>       | Exits all scripts for the session and disconnects the session. |
| <a href="#">Exit_Application</a> | Shuts down the Telnet application.                             |

### Conditionals

| Action                 | Description  |
|------------------------|--|
| <a href="#">If</a>     | Determines which actions to execute.                               |
| <a href="#">If_Not</a> | Determines which actions to stop executing.                        |
| <a href="#">Else</a>   | Starts statements to be executed if an <code>If</code> test fails. |
| <a href="#">End_If</a> | End of statements to be executed for an <code>If</code> test.      |
| <a href="#">While</a>  | Determines which statements to execute.                            |

---

| Action    | Description   |
|-----------|---|
| While_Not | Determines which statements to stop executing.  |
| End_While | End of statements to be executed for a <code>While</code> test.                                     |
| Continue  | Jumps back to the last <code>While</code> statement and re-tests the test value.                    |
| Break     | Jumps to the first statement following the next <code>EndWhile</code> statement (exiting the loop). |

## General Queries

| Action                 | Description  |
|------------------------|--|
| <a href="#">Ask_Ok</a> | Displays a message in a dialog box with an <b>OK</b> button. |

## Send Characters

| Action                              | Description  |
|-------------------------------------|--|
| <a href="#">Keypress_String</a>     | Creates one or more key presses to send the supplied string to the Telnet session. |
| <a href="#">Keypress_Key</a>        | Sends a single keypress to the Telnet session.                                     |
| <a href="#">Scan_String</a>         | Treats the string as scanned data of the type specified.                           |
| <a href="#">Set_Cursor_Position</a> | Moves the cursor to the specified row and column.                                  |

## Message

| Action                        | Description                              |
|-------------------------------|--|
| <a href="#">Message</a>       | Displays a message on the Telnet screen. |
| <a href="#">Message_Clear</a> | Clears the message on the Telnet screen. |

## Sounds

| Action                     | Description  |
|----------------------------|--|
| <a href="#">Beep</a>       | Causes the device to beep.                                       |
| <a href="#">Play_Sound</a> | Causes the device to play the sound specified by the sound name. |

## Waiting

| Action                                 | Description  |
|--|--|
| <a href="#">Wait_For_Screen_Update</a> | Suspends the current script until the screen has been updated.   |
| <a href="#">Delay</a>                  | Suspends the current script until the specified time has passed. |

## Logging

| Action                      | Description   |
|-----------------------------|---|
| <a href="#">Logging_On</a>  | Creates a log file that records all subsequent script execution activity. |
| <a href="#">Logging_Off</a> | Turns off logging for the script.   |

## Call Other Macros

| Action               | Description  |
|----------------------|--|
| <a href="#">Call</a> | Suspends the current script and executes another script. |

## Screen Buttons

| Action   | Description  |
|--|--|
| <a href="#">Button_Bitmap_Create_Emulation</a> | Creates a button with the specified bitmap name using the specified text position.   |
| <a href="#">Button_Bitmap_Create_View</a>      | Creates a button with the specified bitmap name using the specified screen position. |
| <a href="#">Button_Create_Emulation</a>        | Creates a button with the specified text using the specified text position.          |
| <a href="#">Button_Create_View</a>             | Creates a button with the specified text using the specified screen position.        |
| <a href="#">Button_Remove</a>                  | Removes buttons created with the previous actions.                                   |
| <a href="#">Button_Remove_All</a>              | Removes all buttons created with the previous actions.                               |

## Reboot

| Action                 | Description         |
|------------------------|---------------------|
| <a href="#">Reboot</a> | Reboots the device. |

## KeyPress Capture

| Action                                | Description   |
|---------------------------------------|---|
| <a href="#">Keypress_Capture</a>      | Begins a specific key capture and modifier combination.   |
| <a href="#">Keypress_Capture_Stop</a> | Stops the specified key capture and modifier combination. |
| <a href="#">Keyboard_Disable</a>      | Stops all key press captures and modifier combinations    |

## Keyboard

| Action                           | Description             |
|----------------------------------|-------------------------|
| <a href="#">Keyboard_Disable</a> | Disables all keyboards. |

## Blank\_Line

Proceeds to the next instruction without taking any action.

---

### Example

```
Script(Blank_Line_Test)
Activate(From_Menu)
  Comment: This script has some blank lines:
```

```
Return
```

---

### See Also

[Comment](#)



# Goto

Jumps to the supplied label.

---

## Parameters

|              |   |
|--------------|---|
| <i>Label</i> | The label where the script starts running after the <code>Goto</code> . |
|--------------|---|

---

## Format

`Goto: (Label)`

---

## Example

```
Script(Goto_Test)
  Activate(From_Menu)
  Goto: end_script
  Message("Skipped...", 0)
Label: end_script
  Message("The End.", 0)
Return
```

---

## See Also

[Label](#)

# Label

Label to which a `Goto` can jump.

---

## Parameters

|              |  |
|--------------|--|
| <i>Label</i> | Identifies a line in the script for using <code>Goto</code> to change where the script is running. |
|--------------|--|

---

## Format

Label: (Label)

---

## Example

```
Script(Goto_Test)
  Activate(From_Menu)
  Goto: end_script
  Message("Skipped...", 0)
  Label: end_script
  Message("The End.", 0)
  Return
```

---

## See Also

[Goto](#)

## Return

Exits the script normally.

---

### Remarks

If this script was started by another script, the calling script's variables are updated and the calling script resumes.

---

### Example

```
Script (Return_Test)
Activate (From_Menu)
    Comment: This script doesn't do anything
Return
```

---

### See Also

[Abort](#), [Abort\\_All](#), [Disconnect](#), [Exit\\_Application](#)

# Abort

Exits the script immediately.

---

## Remarks

If this script was started by another script, the calling script's variables are not updated and the calling script resumes.

---

## Example

```
Script(Abort_Test)
Activate(From_Menu)
  Comment: This script doesn't do anything.
  Abort
```

---

## See Also

[Return](#), [Abort\\_All](#), [Disconnect](#), [Exit\\_Application](#)

## Abort\_All

Exits all scripts for the session.

---

### Example

```
Script(Abort_All_Test)
Activate(From_Menu)
  Comment: This script causes all of the session's scripts to abort.
  Abort_All
```

---

### See Also

[Return](#), [Abort](#), [Disconnect](#), [Exit\\_Application](#), [Reboot](#)

# Disconnect

Exits all scripts for the session and disconnects the session.

---

## Example

```
Script(Disconnect_Test)
  Activate(From_Menu)
  Comment: This script causes all of the session's
  scripts to end and the session to disconnect.
  Disconnect
```

---

## See Also

[Return](#), [Abort](#), [Abort\\_All](#), [Exit\\_Application](#), [Reboot](#), [Abort](#), [Abort\\_All](#),  
[Exit\\_Application](#), [Suspend](#)

## Exit\_Application

Shuts down the Telnet application.

---

### Parameters

*Return Value*                      The value that the Telnet application returns to the system as it exits.

---

### Format

Exit\_Application (Return Value)

---

### Return Values

The return value is the application exit value Telnet will use when exiting.

---

### Example

```
Script(Exit_Application_Test)
Activate(From_Menu)
  Comment: this script shuts down the Telnet application.
  Exit_Application(1)
```

---

### See Also

[Return](#), [Abort](#), [Abort\\_All](#), [Disconnect](#), [Reboot](#)

## If

If the test is TRUE, the script continues executing until the next `Else` or `EndIf` statement. Otherwise, only executes actions (if any) between the next `Else` and `EndIf` statements.

---

### Parameters

|             |   |
|-------------|---|
| <i>Test</i> | If TRUE, then the next set of actions get executed, up to the <code>Else</code> or <code>End_If</code> , whichever comes first. |
|-------------|---|

---

### Format

```
If (Test)
```

---

### Example

```
Script( If_Test )
Boolean( bOK )
Activate( From_Menu )
    Message_Clear
    bOK = Ask_OK_Cancel( "Press OK to see another message.", "Press OK",
FALSE )
    If( bOK )
        Message( "bOK is TRUE", 0 )
    End_If
Return
```

---

### See Also

[If\\_Not](#), [Else](#), [End\\_If](#)

## If\_Not

If the test is FALSE, the script continues executing until the next `Else` or `EndIf` statement. Otherwise, only executes actions (if any) between the next `Else` and `EndIf` statements.

---

### Parameters

|             |  |
|-------------|--|
| <i>Test</i> | If FALSE, then the next set of actions get executed, up to the <code>Else</code> or <code>End_If</code> , whichever comes first. |
|-------------|--|

---

### Format

```
If_Not (Test)
```

---

### Example

```
Script( If_Not_Test )
Boolean( bOK )
Activate( From_Menu )
    Message_Clear
    bOK = Ask_OK_Cancel( "Press Cancel to see another message.", "Press
Cancel", FALSE )
    If_Not( bOK )
        Message( "bOK is FALSE", 0 )
    End_If
Return
```

---

### See Also

[If](#), [Else](#), [End\\_If](#)

## Else

Start of statements to be executed if an `If` test fails.

---

### Remarks

This command is only valid inside of an `If` block.

---

### Example

```
Script(If_Else_Test)
  Boolean(bOK)
  Activate(From_Menu)
  bOK = Ask_OK_Cancel("Press OK or Cancel.", "Test",
    FALSE)
  If(bOK)
    Message("OK is TRUE", 0)
  Else
    Message("OK is FALSE", 0)
  End_If
  Return
```

---

### See Also

[If](#), [If\\_Not](#), [End\\_If](#)

---

## End\_If

End of statements to be executed for an `If` test.

---

### Example

```
Script( If_Test )
Boolean( bOK )
Activate( From_Menu )
    Message_Clear
    bOK = Ask_OK_Cancel( "Press OK to see another message.", "Press OK",
FALSE )
    If( bOK )
        Message( "bOK is TRUE", 0 )
    End_If
Return
```

---

### See Also

[If, If\\_Not, Else](#)

## While

If the test is TRUE, the statements after `While` and before the next `EndWhile` statement are executed and the `While` statement will be executed again. Otherwise, execution will proceed to the next `EndWhile` statement.

---

### Parameters

*Test* As long as this is TRUE, then the actions up to the `End_While` get executed.

---

### Format

```
While (Test)
```

---

### Remarks

The `While` loop will continue to execute until the test fails, a `Break` command is executed, or the script exits.

---

### Example

```
Script(While_Test)
  Boolean(bOK)
  Activate(From_Menu)
  bOK = TRUE
  While(bOK)
    bOK = Ask_OK_Cancel("Press OK to keep getting this
      message.", "Test", FALSE)
  End_While
  Return
```

---

### See Also

While\_Not, End\_While, Continue, Break

## While\_Not

If the test is FALSE, the statements after `While` and before the next `EndWhile` statement are executed and the `While` statement will be executed again. Otherwise, execution will proceed to the next `EndWhile` statement.

---

### Parameters

*Test* As long as this is FALSE, then the actions up to the `End_While` get executed.

---

### Format

```
While_Not (Test)
```

---

### Remarks

The `While` loop will continue to execute until the test succeeds, a `Break` command is executed, or the script exits.

---

### Example

```
Script( While_Not_Test )
Boolean( bOK )
Activate( From_Menu )
    bOK = FALSE
    While_Not( bOK )
        bOK = Ask_OK_Cancel( "Press Cancel to keep getting this
message.", "Test", FALSE )
    End_While
Return
```

---

### See Also

While, End\_While, Continue, Break

## End\_While

End of statements to be executed for a `While` test.

---

### Example

```
Script(End_While_Test)
  Boolean(bOK)
  Activate(From_Menu)
  bOK = TRUE
  While(bOK)
    bOK = Ask_OK_Cancel("Press OK to keep getting this
      message.", "Test", FALSE)
  End_While
Return
```

---

### See Also

[While](#), [While\\_Not](#), [Continue](#), [Break](#)

# Continue

Jumps back to the last `while` statement and re-tests the test value.

---

## Remarks

This action is only valid inside of a `while` loop.

---

## Example

```
Script(Continue_Test)
  Boolean(bContinue)
  Activate(From_Menu)
  While(TRUE)
    bContinue = Ask_OK_Cancel("Press OK to keep getting
    this message.", "Test", FALSE)
    If(bContinue)
      Continue
    Else
      Break
    End_If
  End_While
Return
```

---

## See Also

[While](#), [While\\_Not](#), [End\\_While](#), [Break](#)

# Break

Jumps to the first statement following the next `EndWhile` statement (exiting the loop).

---

## Remarks

This command is only valid inside of a `While` loop.

---

## Example

```
Script(Continue_Test)
  Boolean(bContinue)
  Activate(From_Menu)
  While(TRUE)
    bContinue = Ask_OK_Cancel("Press OK to keep getting
    this message.", "Test", FALSE)
  If(bContinue)
    Continue
  Else
    Break
  End_If
End_While
Return
```

---

## See Also

[While](#), [While\\_Not](#), [End\\_While](#), [Continue](#)

---

## Ask\_Ok

Displays a message in a box with an **OK** button and waits until the user presses the button.

---

### Parameters

*Message Text*    The message displayed in the box.

*Title Text*        The title of the message box.

---

### Format

```
Ask_Ok(Message Text, "Title Text")
```

---

### Example

```
Script( Test_Ask_Ok )
Activate( From_Menu )
  Message_Clear
  Ask_OK( "Press OK and the script will end.", "Test_Ask_Ok" )
  Return
```

---

### See Also

[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#), [Ask\\_Yes\\_No\\_Cancel](#), [Message](#)

## Keypress\_String

Creates one or more key presses to send the supplied string to the Telnet session.

---

### Parameters

*Characters*                      The text characters to send to the Telnet session.

---

### Format

```
Keypress_String ("Characters")
```

---

### Example

```
Script(Test_Keypress)
  Activate(From_Menu)
  Keypress_String("N")
  Keypress_Key("VT220", "Enter")
  Return
```

---

### See Also

[Keypress\\_Key](#), [Scan\\_String](#), [Set\\_Cursor\\_Position](#)

---

## Keypress\_Key

Sends a single keypress to the Telnet session.

---

### Parameters

|                            |  |
|----------------------------|--|
| <i>Emulation Key Value</i> | The number value of the key to send to the Telnet session. |
|----------------------------|--|

---

### Format

```
Keypress_Key ("Emulation Key Value")
```

---

### Remarks

This is useful for emulation keys that `Keypress_String` cannot handle.

---

### Example

```
Script (Test_Keypress)
  Activate (From_Menu)
  Keypress_String ("N")
  Keypress_Key ("VT220", "Enter")
Return
```

---

### See Also

[Keypress\\_String](#), [Scan\\_String](#), [Set\\_Cursor\\_Position](#)

## Scan\_String

Treats the string as scanned data of the type specified.

---

### Parameters

|                   |   |
|-------------------|---|
| <i>Characters</i> | The string that will be treated as scan data. |
| <i>Scan Type</i>  | The scanner symbology of the scanned data.    |

---

### Format

Scan\_String (Characters, Scan Type)

---

### Remarks

Scanner symbology values can be found in *Symbologies and Values* on page 322.

---

### Example

```
Script(Scan_String_Test)
  String(sScanData)
  Number(nScanType)
  Activate(On_Input, sScanData, nScanType)
  Comment:See if this is a special barcode to indicate
  a keypress.
  Comment: You can adjust the barcode type and test
  strings to suit your purposes.
  If(Number_Equal(nScanType, Get_Scan_Type_Value("CODE
  128")))
    If(String_Equal(sScanData, "Ctrl-A", 0, TRUE))
      Keypress_Key("VT220", "Ctrl-A")
      Return
    End_If
  End_If
  Comment: We didn't use the scan data. Pass it along for
  standard processing.
```

```
Scan_String(sScanData, nScanType)  
Return
```

---

**See Also**

[Keypress\\_String](#), [Keypress\\_Key](#), [Set\\_Cursor\\_Position](#), [Get\\_Scan\\_Type\\_Name](#),  
[Get\\_Scan\\_Type\\_Value](#)

# Set\_Cursor\_Position

Moves the cursor to the specified row and column.

---

## Parameters

|               |   |
|---------------|---|
| <i>Row</i>    | The row where the cursor will go, starting at row 1 for the top row.                |
| <i>Column</i> | The column where the cursor will go, starting at column 1 for the left-most column. |

---

## Format

```
Set_Cursor_Position (Row, Column)
```

---

## Remarks

The top-most row is 1, and the left-most column is 1.

---

## Example

```
Script(Set_Cursor_Position_Test)
  Activate(From_Menu)
  Set_Cursor_Position(15, 6)
  Return
```

---

## See Also

[Keypress\\_String](#), [Keypress\\_Key](#), [Scan\\_String](#), [Get\\_Screen\\_Columns](#), [Get\\_Screen\\_Rows](#), [Get\\_Position\\_Column](#), [Get\\_Position\\_Row](#), [Get\\_Field\\_Index](#), [Get\\_Field\\_Row](#), [Get\\_Field\\_Column](#)

---

# Message

Displays a message on the Telnet screen.

---

## Parameters

|                          |  |
|--------------------------|--|
| <i>Message</i>           | The text that appears in the message box.                            |
| <i>Timeout (Seconds)</i> | After the designated number of seconds until the message is removed. |

---

## Format

```
Message ("Message", Timeout)
```

---

## Remarks

Use 0 for the message box to be displayed until the user selects **OK**.

If the time-out value is greater than 0, the message is removed after that number of seconds elapses.

---

## Example

```
Script(Message_Test)
  Activate(From_Menu)
  Message("Message shows for five seconds", 5)
  Return
```

---

## See Also

[Message\\_Clear](#), [Ask\\_Ok](#)

## Message\_Clear

Clears the message on the Telnet screen.

---

### Example

```
Script(Message_Test)
  Activate(From_Menu)
  Message("Waiting for screen update", 0)
  Wait_For_Screen_Update
  Message_Clear
  Return
```

---

### See Also

[Message](#)

# Beep

Causes the device to beep.

---

## Parameters

|                                |   |
|--------------------------------|---|
| <i>Frequency (Hz)</i>          | The frequency of the beep in cycles per second. |
| <i>Duration (Milliseconds)</i> | The length of the beep.                         |
| <i>Volume</i>                  | The volume at which the beep is played.         |

---

## Format

Beep (Frequency, Duration, Volume)

---

## Remarks

A frequency of 1000 is a good default. The duration is in milliseconds, so a value of 1000 would be one second.

The volume is a value between 0 and 9, where 0 is the softest and 9 is the loudest.

---

## Example

```
Script(Beep_Test)
  Activate(From_Menu)
  Comment: Beep for one-half second at 1000Hz, maximum
  volume:
  Beep(1000, 500, 9)
  Comment: Beep for three seconds at 2500Hz, medium
  volume:
```

```
Beep(2500, 3000, 5)  
Return
```

---

**See Also**

[Play\\_Sound](#), [Speech\\_From\\_Text](#)

# Play\_Sound

Causes the device to play the sound specified by the sound name.

---

## Parameters

*Sound Name*                      The name of the .wav file.

---

## Format

```
Play_Sounds ("Sound Name")
```

---

## Remarks

The sound name may be any .wav file located in the folder specified by the emulation parameters: **Emulation > Sound > Sound Resource Folder**. The sound name may also be any of the sounds in the Resource Editor.

---

## Example

```
Script(Play_Sound_Test)
  Activate(From_Menu)
  Comment: Play the Chimes sound from the Resource Editor.
  Play_Sound("Chimes")
  Return
```

---

## See Also

[Beep](#), [Speech\\_From\\_Text](#)

## Wait\_For\_Screen\_Update

Suspends the current script until the screen has been updated.

---

### Remarks

Any changes to the screen will cause the script to resume; so it is recommended that you put the wait command inside a `While` loop, and only exit the loop once you have detected the screen you want.

`Wait_For_Screen_Update` may be used to wait for a script-created button to be pressed, after the action `Button_Bitmap_Create_Emulation` OR `Button_Create_Emulation`.

---

### Example

```
Script(Wait_For_Screen_Update_Test)
  Activate(From_Menu)
  Message("Waiting for screen update", 0)
  Wait_For_Screen_Update
  Message_Clear
  Return
```

---

### See Also

[Delay](#), [Button\\_Bitmap\\_Create\\_Emulation](#), [Button\\_Bitmap\\_Create\\_View](#),  
[Button\\_Create\\_Emulation](#), [Button\\_Create\\_View](#),  
[Wait\\_For\\_Screen\\_Update\\_With\\_Timeout](#)

# Delay

Suspends the current script until the specified time has passed.

---

## Parameters

*Time (Milliseconds)*      The duration of the delay.

---

## Format

Delay (Time)

---

## Remarks

The time is in milliseconds, so a value of 1000 would be one second.

---

## Example

```
Script(Delay_Test)
  Activate(From_Menu)
  Message("Showing a message for a few seconds.", 0)
  Delay(4000)
  Message_Clear
  Return
```

---

## See Also

[Wait\\_For\\_Screen\\_Update](#), [Wait\\_For\\_Screen\\_Update\\_With\\_Timeout](#), [Get\\_Time](#)

# Logging\_On

Creates a log file that records all subsequent script execution activity.

---

## Parameters

|                           |   |
|---------------------------|---|
| <i>File Path</i>          | The log file path name.                                 |
| <i>Overwrite Previous</i> | Indicates whether the previous log file is overwritten. |

---

## Format

```
Logging_On ("File Path", Overwrite Previous)
```

---

## Remarks

This can be useful while developing a script, but is not recommended for production use.

If *Overwrite Previous* is TRUE, a previous log file will be overwritten. Otherwise, the new information will be appended to the existing file.

Logging is only turned on for the current script. Scripts called by this script will not have logging enabled.

---

## Example

```
Script(Logging_Test)
  Activate(From_Menu)
  Logging_On("TestingLogfile.txt", FALSE)
  Message("A short message, followed by a delay...", 3)
  Delay(1500)
```

Logging\_Off  
Return

---

**See Also**

[Logging\\_Off](#), [Get\\_MAC\\_Address](#), [Get\\_IP\\_Address](#), [Get\\_Workstation\\_ID](#),  
[Get\\_Session\\_Number](#)

## Logging\_Off

Turns off logging for the script.

---

### Example

```
Script( Logging_Off_Test )
Activate( From_Menu )
  Logging_On( "TestingLogfile.txt", FALSE )
  Message( "A short message, followed by a delay...", 3 )
  Delay( 1500 )
  Logging_Off
  Message( "This message does not get logged", 3 )
Return
```

---

### See Also

[Logging\\_On](#)

# Call

Suspends the current script and executes another script. The current script resumes when the called script exits.

---

## Parameters

*Script Name*                      The name of the script to call.

---

## Format

Call: Script Name

---

## Remarks

The name of the script is case-sensitive.

Refer to the *Wavelink Telnet Client Scripting Reference Guide* for more information about script nesting.

---

## Example

```
Script (Messenger)
  String (strMessage)
  Message (strMessage, 0)
  Return
Script (Call_Test)
  String (MessageToUse)
  Activate (From_Menu)
  MessageToUse = "Calling Another Script"
  Call: Messenger
  strMessage <---MessageToUse
  Return
```

---

## See Also

[Abort](#), [Abort\\_All](#), [Return](#)

---

## Button\_Bitmap\_Create\_Emulation

Creates a button with the specified bitmap name, and puts the left side of it where emulation text at the supplied coordinates would be.

---

### Parameters

|                    |  |
|--------------------|--|
| <i>Bitmap Name</i> | The name of the bitmap.                                    |
| <i>Row</i>         | The top of the bitmap starts in this text row.             |
| <i>Column</i>      | The left side of the bitmap starts in this text column.    |
| <i>Pressed</i>     | Indicates whether the button has been pressed by the user. |

---

### Format

```
Button_Bitmap_Create_Emulation ("Bitmap Name", Row, Column, Pressed)
```

---

### Remarks

You can add bitmaps to the resource file by using the Resource Editor. Refer to *Wavelink Telnet Client User Guide* for more information. Row-1 is the top line of text on the screen; column-1 is the left-most column of text on the screen.

Each time the button is pressed, the boolean variable specified will be set to TRUE. You will need to reset the variable if you want to detect future button presses. All buttons created by the script will be removed when the script exits. The `Wait_For_Screen_Update` action can be used to wait for a button to be pressed.

The button will be hidden if the emulation text at that location is hidden.

---

### Example

```
Script( Button_Bitmap )
Boolean( Pressed )
Activate( From_Menu )
    Button_Bitmap_Create_Emulation( "GOCONTROL", 6, 17, Pressed )
    While_Not( Pressed )
        Wait_For_Screen_Update
    End_While
    Button_Remove( "GOCONTROL" )
    Pressed = FALSE
Return
```

---

## See Also

[Wait\\_For\\_Screen\\_Update](#), [Button\\_Bitmap\\_Create\\_View](#), [Button\\_Create\\_Emulation](#),  
[Button\\_Create\\_View](#), [Button\\_Remove](#), [Button\\_Remove\\_All](#)

---

## Button\_Bitmap\_Create\_View

Creates a button with the specified bitmap name. This command is the same as [Button\\_Bitmap\\_Create\\_Emulation](#), except that the screen position is used instead of text position, allowing the button to always be visible.

---

### Parameters

|                    |  |
|--------------------|--|
| <i>Bitmap Name</i> | The name of the bitmap.                                    |
| <i>Row</i>         | The top of the bitmap starts in this text row.             |
| <i>Column</i>      | The left side of the bitmap starts in this text column.    |
| <i>Pressed</i>     | Indicates whether the button has been pressed by the user. |

---

### Format

```
Button_Bitmap_Create_View ("Bitmap Name", Row, Column, Pressed)
```

---

### Remarks

If `Button_Bitmap_Create_View` is used to create a button at position 1,1, that button will always be in the upper-left corner of the Telnet view screen. A `Button_Bitmap_Create_Emulation` button will be hidden if the emulation text at that location is hidden.

A bottom and/or right value of 1000 represents the bottom or right side of the screen. For example, a button at position 1,990 would start 11 columns left of the upper-right corner of the screen.

Each time the button is pressed, the boolean variable specified will be set to TRUE. You will need to reset the variable if you want to detect future button

presses. The `Wait_For_Screen_Update` action can be used to wait for a button to be pressed. All buttons created by the script will be removed when the script exits.

---

## Example

```
Script( Button_Bitmap_Create_View_Test )
Boolean( Pressed )
Activate( From_Menu )
    Button_Bitmap_Create_View( "STOPCONTROL", 1000, 1000, Pressed )
    While_Not( Pressed )
        Wait_For_Screen_Update
    End_While
    Button_Remove_All
Return
```

---

## See Also

[Button\\_Bitmap\\_Create\\_Emulation](#), [Button\\_Create\\_Emulation](#), [Button\\_Create\\_View](#), [Button\\_Remove](#), [Button\\_Remove\\_All](#)

---

## Button\_Create\_Emulation

Creates a button with the specified text and puts the left side of it where emulation text at the supplied coordinates would be.

---

### Parameters

|                |  |
|----------------|--|
| <i>Text</i>    | The text displayed in the button.                          |
| <i>Row</i>     | The top of the bitmap starts in this text row.             |
| <i>Column</i>  | The left side of the bitmap starts in this text column.    |
| <i>Width</i>   | The number of characters in the button text.               |
| <i>Pressed</i> | Indicates whether the button has been pressed by the user. |

---

### Format

```
Button_Create_Emulation ("Text", Row, Column, Width, Pressed)
```

---

### Remarks

If the width value is 0, the button will be sized to fit the text. Each time the button is pressed, the boolean variable specified will be set to TRUE. You will need to reset the variable if you want to detect future button presses. All buttons created by the script will be removed when the script exits. The `Wait_For_Screen_Update` action can be used to wait for a button to be pressed.

The button will be hidden if emulation text at that location is hidden.

---

### Example

```
Script( Button_Create_Emulation_Test )
Boolean( Pressed )
Activate( From_Menu )
    Button_Create_Emulation( "Emulation Button", 2, 1, 0, Pressed )
    While_Not( Pressed )
        Wait_For_Screen_Update
    End_While
    Button_Remove( "Emulation Button" )
    Pressed = FALSE
Return
```

---

## See Also

[Wait\\_For\\_Screen\\_Update](#), [Button\\_Create\\_View](#), [Button\\_Remove](#), [Button\\_Remove\\_All](#),  
[Button\\_Bitmap\\_Create\\_Emulation](#), [Button\\_Bitmap\\_Create\\_View](#)

---

## Button\_Create\_View

Creates a button with the specified text. This command is the same as [Button\\_Create\\_Emulation](#) except that the screen position is used instead of the text position, allowing the button to always be visible.

---

### Parameters

|                |  |
|----------------|--|
| <i>Text</i>    | The text displayed in the button.                          |
| <i>Row</i>     | The top of the bitmap starts in this text row.             |
| <i>Column</i>  | The left side of the bitmap starts in this text column.    |
| <i>Width</i>   | The number of characters in the button text.               |
| <i>Pressed</i> | Indicates whether the button has been pressed by the user. |

---

### Format

```
Button_Create_View ("Text", Row, Column, Width, Pressed)
```

---

### Remarks

If `Button_Create_View` is used to create a button at position 1,1, that button will always be in the upper-left corner of the Telnet view screen. A `Button_Create_Emulation` button will be hidden if the emulation text at that location is hidden. A bottom and/or right value of 1000 represents the bottom or right side of the screen. For example, a button at position 1,900 would start 11 columns left of the upper-right corner of the screen.

Each time the button is pressed, the boolean variable specified will be set to TRUE. You will need to reset the variable if you want to detect future button presses. The

`Wait_For_Screen_Update` action can be used to wait for a button to be pressed. All buttons created by the script will be removed when the script exits.

---

## Example

```
Script( Button_Create_View_Test )
Boolean( Pressed )
Activate( From_Menu )
    Button_Create_View( "This is a button", 1, 1, 0, Pressed )
    While_Not( Pressed )
        Wait_For_Screen_Update
    End_While
    Button_Remove_All
Return
```

---

## See Also

[Button\\_Create\\_Emulation](#), [Button\\_Remove](#), [Button\\_Remove\\_All](#),  
[Button\\_Bitmap\\_Create\\_Emulation](#), [Button\\_Bitmap\\_Create\\_View](#)



## Button\_Remove\_All

Removes all buttons created with the `Button_Create_Emulation` and `Button_Create_View` action for this script.

---

### Example

```
Script( Button_Remove_All_Test )
Boolean( Pressed )
Activate( From_Menu )
    Button_Create_View( "This is a button", 1, 1, 0, Pressed )
    Button_Create_View( "Another button", 4, 1, 0, Pressed )
    While_Not( Pressed )
        Wait_For_Screen_Update
    End_While
    Button_Remove_All
    Ask_OK( "All buttons removed.", "Button_Remove_All_Test" )
Return
```

---

### See Also

[Button\\_Create\\_Emulation](#), [Button\\_Create\\_View](#), [Button\\_Remove](#),  
[Button\\_Bitmap\\_Create\\_Emulation](#), [Button\\_Bitmap\\_Create\\_View](#)

# Reboot

Reboots the device. Any subsequent commands will not be executed unless the reboot fails.

---

## Parameters

*Cold Boot* Indicates whether the device will cold boot.

---

## Format

Reboot (Cold Boot)

If Cold Boot is TRUE, the device will cold boot. Some applications and settings may be lost.

If Cold Boot is FALSE, the device will not perform a cold boot.

---

**NOTE** Cold boot is only supported by some mobile devices.

---

---

## Example

```
Script (Reboot_Test)
  Activate (From_Menu)
  Comment: Do a regular reboot of the device, not a cold boot.
  Reboot (FALSE)
  Return
```

---

## See Also

[Ask\\_Yes\\_No](#), [Disconnect](#), [Exit\\_Application](#), [Abort\\_All](#), [Suspend](#), [Get\\_Time\\_Since\\_Reset](#)

# Keypress\_Capture

Begins a keypress capture.

---

## Parameters

|               |  |
|---------------|--|
| <i>Value1</i> | The Key Value (integer).               |
| <i>Value1</i> | The modifier (Shift, Ctrl, Alt, None). |
| <i>Value1</i> | A Boolean variable.                    |

---

## Format

```
Keypress_Capture(Value1, "Value2", Value3)
```

---

## Remarks

The Key Value is the Diagnostics Windows Keyboard Test value for the keypress (for example, 0079 would become 0x0079).

Supported modifiers are Shift, Ctrl, Alt and None. An empty string is treated as None.

When the specified key and modifier combination is pressed, the boolean variable will be set to TRUE.

---

## Example

```
Script( Ctrl-F10_Keypress_Test )
Boolean( bKeyPressed )
Boolean( bF1Pressed )
Activate( From_Menu )
    Message( "F1 is captured; press control-F10 to exit", 0 )
    Keypress_Capture( 0x79, "Ctrl", bKeyPressed )
    Keypress_Capture( 0x70, "", bF1Pressed )
    While_Not( bKeyPressed )
        Wait_For_Screen_Update
```

```
    If( bF1Pressed )
        bF1Pressed = FALSE
        Keypress_Capture_Stop( 0x70, "" )
        Message( "F1 Pressed; capturing stopped for F1.", 3 )
    End_If
End_While
Keypress_Capture_Stop( 0x79, "C" )
Keypress_Capture_Stop_All
Message( "Ctrl-F10 Pressed - script is done", 5 )
Return
```

# Keypress\_Capture\_Stop

Stops capturing the specified key and modifier combination.

---

## Parameters

|               |  |
|---------------|--|
| <i>Value1</i> | The Key Value (integer).               |
| <i>Value2</i> | The modifier (Shift, Ctrl, Alt, None). |

---

## Format

```
Keypress_Capture_Stop (Value1, "Value2")
```

---

## Remarks

Supported modifiers are Shift, Ctrl, Alt and None. An empty string is treated as None."

---

## Example

```
Script( Ctrl-F10_Keypress_Test )
Boolean( bKeyPressed )
Boolean( bF1Pressed )
Activate( From_Menu )
    Message( "F1 is captured; press control-F10 to exit", 0 )
    Keypress_Capture( 0x79, "Ctrl", bKeyPressed )
    Keypress_Capture( 0x70, "", bF1Pressed )
    While_Not( bKeyPressed )
        Wait_For_Screen_Update
        If( bF1Pressed )
            bF1Pressed = FALSE
            Keypress_Capture_Stop( 0x70, "" )
            Message( "F1 Pressed; capturing stopped for F1.", 3 )
        End_If
    End_While
Keypress_Capture_Stop( 0x79, "C" )
Keypress_Capture_Stop_All
```

```
Message( "Ctrl-F10 Pressed - script is done", 5 )  
Return
```

## Keypress\_Capture\_Stop\_All

Stops capturing all key and modifier combinations.

---

### Example

```
Script( Ctrl-F10_Keypress_Test )
Boolean( bKeyPressed )
Boolean( bF1Pressed )
Activate( From_Menu )
    Message( "F1 is captured; press control-F10 to exit", 0 )
    Keypress_Capture( 0x79, "Ctrl", bKeyPressed )
    Keypress_Capture( 0x70, "", bF1Pressed )
    While_Not( bKeyPressed )
        Wait_For_Screen_Update
        If( bF1Pressed )
            bF1Pressed = FALSE
            Keypress_Capture_Stop( 0x70, "" )
            Message( "F1 Pressed; capturing stopped for F1.", 3 )
        End_If
    End_While
    Keypress_Capture_Stop( 0x79, "C" )
    Keypress_Capture_Stop_All
    Message( "Ctrl-F10 Pressed - script is done", 5 )
Return
```

---

# Keyboard\_Disable

Disables the keyboards.

---

## Parameters

*Value1*                                      Boolean (Disable)

---

## Format

Keyboard\_Disable(Value1)

---

## Remarks

If Disable is TRUE, the hardware and on-screen keyboards for the session are disabled or the session is disconnected.

The keyboards will remain disabled until a script calls this action with a FALSE Disable value.

---

## Example

```
Script( Keyboard_Disable_Test )
Activate( From_Menu )
    Keyboard_Disable( TRUE )
    Message( "Keyboard is disabled for a few seconds", 0 )
    Delay( 3000 )
    Keyboard_Disable( FALSE )
    Message( "Keyboard is enabled", 3 )
Return
```

## Boolean Values

This section contains a list of actions that return boolean values. The following action categories are described in this section:

### Boolean Assignments

| Action                      | Description   |
|-----------------------------|---|
| <a href="#">Boolean_Set</a> | Set the value of a Boolean variable.  |
| <a href="#">Boolean_Not</a> | Set the value of a Boolean variable to FALSE if the parameter is TRUE. Set the value to TRUE if the parameter is FALSE. |
| <a href="#">Boolean_And</a> | Test each of the parameters and return TRUE if all are TRUE or FALSE if one or more are FALSE.                          |
| <a href="#">Boolean_Or</a>  | Test each of the parameters and return TRUE if one or more are true or FALSE if all are FALSE.                          |

### Boolean Comparisons

| Action                            | Description   |
|-----------------------------------|---|
| <a href="#">Boolean_Equal</a>     | Compare the two parameters and return TRUE if they are both TRUE or if they are both FALSE. |
| <a href="#">Boolean_Not_Equal</a> | Compare the two parameters and return TRUE if they do not have the same value.              |

### String Comparisons

| Action                                    | Description   |
|---|---|
| <a href="#">String_Empty</a>              | Check the length of the string to determine if it's an empty string.  |
| <a href="#">String_Less_Than</a>          | Compare the two strings and determine their alphabetical order.   |
| <a href="#">String_Less_Than_Or_Equal</a> | Compare the two strings and determine whether one precedes the other in alphabetical order, or if they are the same string. |
| <a href="#">String_Equal</a>              | Compare the two strings and determine if they are both TRUE or are both FALSE.  |

---

| <b>Action</b>                                | <b>Description</b>  |
|--|---|
| <a href="#">String_Greater_Than_Or_Equal</a> | Compare the two strings and determine whether one follows the other in alphabetical order, or they are the same string. |
| <a href="#">String_Greater_Than</a>          | Compare the two strings and determine whether one follows the other in alphabetical order.                              |
| <a href="#">String_Not_Equal</a>             | Compare the two strings and return TRUE if they do not have the same value.   |

## Field Identifiers and Data

| Action                                       | Description   |
|--|---|
| <a href="#">Set_Field_Data_ID</a>            | Sets the Data ID for a field.   |
| <a href="#">Set_Field_Symbology_ID</a>       | Sets the Symbology ID for a field.                                    |
| <a href="#">Get_Field_Symbology_Operator</a> | Query whether the field data matches the Data ID and/or Symbology ID. |
| <a href="#">Set_Field_Append_Scan_Data</a>   | Controls whether to append scan data in the field.                    |
| <a href="#">Set_Field_Com_Data_Field</a>     | Sets a field to be the Com Data Field for the screen.                 |
| <a href="#">Set_Field_Prefix_Scan_Data</a>   | Sets the data prefixed to a field when the field is scanned.          |
| <a href="#">Get_Field_Append_Scan_Data</a>   | Query whether data is appended when the field is scanned.             |

## Integer Comparison

| Action                                       | Description  |
|--|--|
| <a href="#">Number_Less_Than</a>             | Compares two numbers and determines if one is less than the other.                         |
| <a href="#">Number_Less_Than_Or_Equal</a>    | Compares two numbers and determines if one is less than the other or if they are the same. |
| <a href="#">Number_Equal</a>                 | Compare two numbers and determine whether they are equal.                                  |
| <a href="#">Number_Greater_Than_Or_Equal</a> | Compare two numbers and determine if one is greater than the other or if they are equal.   |
| <a href="#">Number_Greater_Than</a>          | Compare two numbers and determine if one is greater than the other.                        |
| <a href="#">Number_Not_Equal</a>             |  |

## General Queries

| Action  | Description  |
|---|--|
| <a href="#">Ask_OK_Cancel</a>                       | Displays a message and waits until the user selects a button.  |
| <a href="#">Ask_Yes_No</a>                          | Displays a message and waits until the user selects a button.  |
| <a href="#">Wait_For_Screen_Update_With_Timeout</a> | Suspends the current script until the screen has been updated. |

## Suspend

| Action                  | Description          |
|-------------------------|----------------------|
| <a href="#">Suspend</a> | Suspends the device. |

## Search the Screen

| Action                        | Description                                |
|-------------------------------|--|
| <a href="#">Search_Screen</a> | Searches the screen for the supplied text. |

## WEB Emulation Commands

| Action                                 | Description   |
|--|---|
| <a href="#">Web_Navigate</a>           | Navigates WEB emulation to the URL provided.                            |
| <a href="#">Web_Navigate_Frame</a>     | Navigates WEB emulation to the URL provided within the indicated frame. |
| <a href="#">Web_Navigate_Post_Data</a> | Navigates WEB emulation to the URL provided.                            |
| <a href="#">Web_Scripting</a>          | Instructs WEB emulation to execute the scripting information.           |
| <a href="#">Web_Search_Source</a>      | Searches the page source of the current WEB emulation page.             |

## Speech Commands

| Action                                     | Description  |
|--|--|
| <a href="#">Speech_From_Text_Available</a> | Determines whether text-to-speech is supported.  |
| <a href="#">Speech_From_Text</a>           | Converts text into sound and plays it on the computer.   |
| <a href="#">Speech_To_Text_Available</a>   | Determines whether speech-to-text is supported.  |
| <a href="#">Speech_To_Text</a>             |  |
| <a href="#">Speech_To_Text_No_Wait</a>     | Listens to the user speak and returns the text equivalent.                                       |
| <a href="#">Speech_To_Text_Cancel</a>      | Provides a way for the script to perform other functions while the speech-to-text action occurs. |

| Action  | Description  |
|---|--|
| <a href="#">Speech_Setting_Available</a>        | Identifies speech settings by case-insensitive name strings.                                     |
| <a href="#">Speech_Change_Setting</a>           | Changes the speech setting to the specified value.   |
| <a href="#">Speech_Get_Setting</a>              | Gets the value of the speech setting.  |
| <a href="#">Speech_Get_Setting_Max</a>          | Gets the largest value for the speech setting.   |
| <a href="#">Speech_Find_Setting_Value</a>       | Searches all possible value descriptions for the speech setting.                                 |
| <a href="#">Speech_Get_Setting_Value_Desc</a>   | Gets a description of the speech setting value.  |
| <a href="#">Speech_To_Text_Get_User_Name</a>    | Gets the user name.  |
| <a href="#">Speech_To_Text_Change_User_Name</a> | Changes the user name being used by the speech-to-text engine.                                   |
| <a href="#">Speech_From_Text_Error_Desc</a>     | Gets an error description for the last speech-from-text action.                                  |
| <a href="#">Speech_To_Text_Error_Desc</a>       | Gets an error description for the last speech-to-text action.                                    |
| <a href="#">Speech_From_Text_Cancel</a>         | Provides a way for the script to perform other functions while the text-to-speech action occurs. |

## Printer Commands

| Action                         | Description  |
|--------------------------------|--|
| <a href="#">Printer_Data</a>   | Sends data directly to the currently defined printer.                    |
| <a href="#">Printer_Repeat</a> | Instructs the printer to reprint the last item printed.                  |
| <a href="#">Printer_Cancel</a> | Instructs the printer to discard all Printer_Data data already received. |

---

# Boolean\_Set

Set the value of a Boolean variable.

---

## Parameters

|             |   |
|-------------|---|
| <i>Test</i> | May be a Boolean action, variable, or constraint. |
|-------------|---|

---

## Format

```
Boolean_Set (Test)
```

---

## Return Value

Returns a Boolean. TRUE if the test is TRUE, returns FALSE otherwise.

---

## Remarks

A typical use of `Boolean_Set` is to set a variable to the return value of another action. The equal sign (=) is the symbol for `Boolean_Set` in the Script Editor.

---

## Example

```
Script(Boolean_Set_Test)
Boolean(bResult)
Activate(From_Menu)
    bResult = Ask_OK_Cancel("OK for TRUE, Cancel for FALSE",
"Boolean_Set_Test", FALSE)
    If(bResult)
        Message("bResult is TRUE", 5)
    Else
        Message("bResult is FALSE", 5)
```

End\_If  
Return

---

**See Also**

[Boolean\\_Not](#), [Boolean\\_And](#), [Boolean\\_Or](#), [Boolean\\_Equal](#), [Boolean\\_Not\\_Equal](#),  
[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#)

---

## Boolean\_Not

Set the value of a Boolean variable to FALSE if the parameter is TRUE. Set the value of a Boolean variable to TRUE if the parameter is FALSE.

---

### Parameters

|             |   |
|-------------|---|
| <i>Test</i> | May be a Boolean action, variable, or constraint. |
|-------------|---|

---

### Format

`Boolean_Not (Test)`

---

### Return Value

Returns a Boolean. FALSE if the test is TRUE, returns TRUE otherwise.

---

### Example

```
Script(Boolean_Not_Test)
Boolean(bResult)
Activate(From_Menu)
  bResult = Ask_OK_Cancel("OK for FALSE, Cancel for TRUE",
"Boolean_Not_Test", FALSE)
  bResult = Boolean_Not(bResult)
  If(bResult)
    Message("bResult is TRUE", 5)
  Else
    Message("bResult is FALSE", 5)
  End_If
Return
```

---

### See Also

Boolean\_Set, Boolean\_And, Boolean\_Or, Boolean\_Equal, Boolean\_Not\_Equal,  
Ask\_OK\_Cancel, Ask\_Yes\_No

## Boolean\_And

Test each of the parameters and return TRUE if all parameters are TRUE. Return FALSE if one or more parameters are FALSE. One to five parameters may be used for this action.

---

### Parameters

|              |  |
|--------------|--|
| <i>Test1</i> | A Boolean variable, constant, or action.           |
| <i>Test2</i> | An optional Boolean variable, constant, or action. |
| <i>Test3</i> | An optional Boolean variable, constant, or action. |
| <i>Test4</i> | An optional Boolean variable, constant, or action. |
| <i>Test5</i> | An optional Boolean variable, constant, or action. |

---

### Format

`Boolean_And (Test1, Test2, Test3, etc.)`

---

### Return Value

Returns a Boolean. TRUE if all test values are TRUE; returns FALSE otherwise.

---

### Remarks

All tests will be evaluated each time this action is taken. Use Boolean variables as the parameters instead of actions to make the script easier to read and understand.

---

### Example

```
Script(Boolean_And_Test)
Boolean(bResultAll)
Boolean(bResult1)
Boolean(bResult2)
Activate(From_Menu)
    bResult1 = Ask_OK_Cancel("Hit OK in every message box", "Message 1",
FALSE)
    bResult2 = Ask_OK_Cancel("Hit OK again", "Message 2", FALSE)
    bResultAll = Boolean_And(bResult1, bResult2)
    If(bResultAll)
        Message("bResultAll is TRUE", 5)
    Else
        Message("bResultAll is FALSE", 5)
    End_If
Return
```

---

## See Also

[Boolean\\_Set](#), [Boolean\\_Not](#), [Boolean\\_Or](#), [Boolean\\_Equal](#), [Boolean\\_Not\\_Equal](#),  
[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#)

---

## Boolean\_Or

Test each of the parameters and return TRUE if one or more parameters are TRUE. Return FALSE if all parameters are FALSE. One to five parameters may be used for this action.

---

### Parameters

|              |  |
|--------------|--|
| <i>Test1</i> | A Boolean variable, constant, or action.           |
| <i>Test2</i> | An optional Boolean variable, constant, or action. |
| <i>Test3</i> | An optional Boolean variable, constant, or action. |
| <i>Test4</i> | An optional Boolean variable, constant, or action. |
| <i>Test5</i> | An optional Boolean variable, constant, or action. |

---

### Format

`Boolean_Or (Test1, Test2, Test 3, etc.)`

---

### Return Value

Returns a Boolean. TRUE if one or more test values are TRUE. Returns FALSE otherwise.

---

### Remarks

All tests will be evaluated each time this action is taken. Use Boolean variables as the parameters instead of actions to make the script easier to read and understand.

---

## Example

```
Script(Boolean_Or_Test)
Boolean(bResultAll)
Boolean(bResult1)
Boolean(bResult2)
Activate(From_Menu)
    bResult1 = Ask_OK_Cancel("Hit OK in one message box", "Message 1",
FALSE)
    bResult2 = Ask_OK_Cancel("Hit OK if you hit Cancel in the last box",
"Message 2", FALSE)
    bResultAll = Boolean_Or(bResult1, bResult2)
    If(bResultAll)
        Message("bResultAll is TRUE", 5)
    Else
        Message("bResultAll is FALSE", 5)
    End_If
Return
```

---

## See Also

[Boolean\\_Set](#), [Boolean\\_Not](#), [Boolean\\_And](#), [Boolean\\_Equal](#), [Boolean\\_Not\\_Equal](#),  
[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#)



```
bResultAll = Boolean_Equal(bResult1, bResult2)
If(bResultAll)
    Message("Both responses were the same", 5)
Else
    Message("The responses were different", 5)
End_If
Return
```

---

## **See Also**

[Boolean\\_Not\\_Equal](#), [Boolean\\_Set](#), [Boolean\\_Not](#), [Boolean\\_And](#), [Boolean\\_Or](#),  
[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#)

---

## Boolean\_Not\_Equal

Compare the two parameters and return TRUE if they do not have the same value.

---

### Parameters

*Test1* A Boolean variable, constant, or action.

*Test2* A Boolean variable, constant, or action.

---

### Format

```
Boolean_Not_Equal (Test1, Test2)
```

---

### Return Value

Returns a Boolean. FALSE if both Test1 and Test2 are TRUE, or both Test1 and Test2 are FALSE. Returns TRUE otherwise.

---

### Remarks

Use Boolean variables as the parameters instead of actions to make the script easier to read and understand.

---

### Example

```
Script(Boolean_Not_Equal_Test)
Boolean(bResultAll)
Boolean(bResult1)
Boolean(bResult2)
Activate(From_Menu)
    bResult1 = Ask_Yes_No("Hit Yes in one message box", "Message 1", FALSE)
    bResult2 = Ask_Yes_No("Hit No if you hit Yes in the last message box",
"Message 2", FALSE)
    bResultAll = Boolean_Not_Equal(bResult1, bResult2)
```

```
If (bResultAll)
    Message("The responses were different", 5)
Else
    Message("Both responses were the same", 5)
End_If
Return
```

---

## See Also

[Boolean\\_Equal](#), [Boolean\\_Set](#), [Boolean\\_Not](#), [Boolean\\_And](#), [Boolean\\_Or](#), [Ask\\_OK\\_Cancel](#),  
[Ask\\_Yes\\_No](#)

---

## String\_Empty

Check the length of the string to determine if it is an empty string.

---

### Parameters

*String* Check if this string is empty.

---

### Format

String\_Empty (String)

---

### Return Value

Returns Boolean. TRUE if the string is 0 characters in length, FALSE otherwise.

---

### Example

```
Script(String_Empty_Test)
String(strEntered)
Boolean(bEmpty)
Activate(From_Menu)
    strEntered = Ask_String("Type a string", "String_Empty_Test", 0, 0, "")
    bEmpty = String_Empty(strEntered)
    If(bEmpty)
        Message("String is empty.", 5)
    Else
        Message("String is not empty.", 5)
    End_If
Return
```

---

### See Also

[String\\_Less\\_Than](#), [String\\_Less\\_Than\\_Or\\_Equal](#), [String\\_Equal](#),  
[String\\_Greater\\_Than\\_Or\\_Equal](#), [String\\_Greater\\_Than](#), [String\\_Not\\_Equal](#), [String\\_Set](#),

Number\_To\_String\_Decimal, Ask\_String, Get\_Screen\_Text, Speech\_To\_Text,  
Get\_Scan\_Type\_Name

---

## String\_Less\_Than

Compare the two strings and determine their alphabetical order.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Test1</i>          | Gets compared with Test2.  |
| <i>Test2</i>          | Gets compared with Test1.  |
| <i>Maximum Length</i> | Indicates whether the number of characters is limited.                 |
| <i>Ignore Case</i>    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
String_Less_Than ("Test1", "Test2", Maximum Length, Ignore Case)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 precedes Test2 in alphabetical ordering, FALSE otherwise.

---

### Remarks

If the Maximum Length value is greater than 0, any characters after the specified number of characters are ignored. If Ignore Case is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script(String_Less_Than_Test)  
String(strEntered1)
```

```
String(strEntered2)
Boolean(bLessThan)
Activate(From_Menu)
    strEntered1 = Ask_String("Type the first string",
        "String_Less_Than_Test", 0, 0, "")
    strEntered2 = Ask_String("Type the second string",
        "String_Less_Than_Test", 0, 0, "")
    bLessThan = String_Less_Than(strEntered1, strEntered2, 0, TRUE)
    If(bLessThan)
        Message("First string is less than second string", 5)
    Else
        Message("First string is not less than second string", 5)
    End_If
Return
```

---

### **See Also**

[String\\_Empty](#), [String\\_Less\\_Than\\_Or\\_Equal](#), [String\\_Equal](#),  
[String\\_Greater\\_Than\\_Or\\_Equal](#), [String\\_Greater\\_Than](#), [String\\_Not\\_Equal](#), [String\\_Set](#),  
[Number\\_To\\_String\\_Decimal](#), [Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#),  
[Get\\_Scan\\_Type\\_Name](#)

---

## String\_Less\_Than\_Or\_Equal

Compare the two strings and determine whether one precedes the other in alphabetical order, or they are the same string.

---

### Parameters

|                |  |
|----------------|--|
| Test1          | Gets compared with Test2.  |
| Test2          | Gets compared with Test1.  |
| Maximum Length | Indicates whether the number of characters is limited.                 |
| Ignore Case    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
String_Less_Than_Or_Equal ("Test1", "Test2", Maximum Length, Ignore Case)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 precedes Test2 in alphabetical ordering or they are the same string, FALSE otherwise.

---

### Remarks

If the `Maximum Length` value is greater than 0, any characters after the specified number of characters are ignored. If `Ignore Case` is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script(String_Less_Than_Or_Equal_Test)
String(strEntered1)
String(strEntered2)
Boolean(bLessThan)
Activate(From_Menu)
    strEntered1 = Ask_String("Type the first string",
        "String_Less_Than_Or_Equal_Test", 0, 0, "")
    strEntered2 = Ask_String("Type the second string",
        "String_Less_Than_Or_Equal_Test", 0, 0, "")
    bLessThan = String_Less_Than_Or_Equal(strEntered1,
        strEntered2, 0, TRUE)
    If(bLessThan)
        Message("First string is less than or equal to second
            string", 5)
    Else
        Message("First string is not less than or equal to
            second string", 5)
    End_If
Return
```

---

## See Also

[String\\_Empty](#), [String\\_Less\\_Than](#), [String\\_Equal](#), [String\\_Greater\\_Than\\_Or\\_Equal](#),  
[String\\_Greater\\_Than](#), [String\\_Not\\_Equal](#), [String\\_Set](#), [Number\\_To\\_String\\_Decimal](#),  
[Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#), [Get\\_Scan\\_Type\\_Name](#)

---

## String\_Equal

Compare the two strings and determine if they are both TRUE or are both FALSE.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Test1</i>          | Gets compared with Test2.  |
| <i>Test2</i>          | Gets compared with Test1.  |
| <i>Maximum Length</i> | Indicates whether the number of characters is limited.                 |
| <i>Ignore Case</i>    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
String_Equal ("Test1", "Test2", Maximum Length, Ignore Case)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 precedes Test2 are the same string, FALSE otherwise.

---

### Remarks

If the *Maximum Length* value is greater than 0, any characters after the specified number of characters are ignored. If *Ignore Case* is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script(String_Equal_Test)  
String(strEntered1)
```

```
String(strEntered2)
Boolean(bEqual)
Activate(From_Menu)
    strEntered1 = Ask_String("Type the first string",
        "String_Equal_Test", 0, 0, "")
    strEntered2 = Ask_String("Type the second string",
        "String_Equal_Test", 0, 0, "")
    bEqual = String_Equal(strEntered1, strEntered2, 0, TRUE)
    If(bEqual)
        Message("First string is equal to second string", 5)
    Else
        Message("First string is not equal to second string", 5)
    End_If
Return
```

---

## See Also

[String\\_Empty](#), [String\\_Less\\_Than](#), [String\\_Less\\_Than\\_Or\\_Equal](#),  
[String\\_Greater\\_Than\\_Or\\_Equal](#), [String\\_Greater\\_Than](#), [String\\_Not\\_Equal](#), [String\\_Set](#),  
[Number\\_To\\_String\\_Decimal](#), [Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#),  
[Get\\_Scan\\_Type\\_Name](#)

---

## String\_Greater\_Than\_Or\_Equal

Compare the two strings and determine whether one follows the other in alphabetical order, or they are the same string.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Test1</i>          | Gets compared with Test2.  |
| <i>Test2</i>          | Gets compared with Test1.  |
| <i>Maximum Length</i> | Indicates whether the number of characters is limited.                 |
| <i>Ignore Case</i>    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
String_Greater_Than_Or_Equal ("Test1", "Test2", Maximum Length, Ignore Case)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 precedes Test2 in alphabetical ordering or they are the same string, FALSE otherwise.

---

### Remarks

If the *Maximum Length* value is greater than 0, any characters after the specified number of characters are ignored. If *Ignore Case* is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script(String_Greater_Than_Or_Equal_Test)
String(strEntered1)
String(strEntered2)
Boolean(bGreaterThan)
Activate(From_Menu)
    strEntered1 = Ask_String("Type the first string",
        "String_Greater_Than_Or_Equal_Test", 0, 0, "")
    strEntered2 = Ask_String("Type the second string",
        "String_Greater_Than_Or_Equal_Test", 0, 0, "")
    bGreaterThan = String_Greater_Than_Or_Equal(strEntered1,
        strEntered2, 0, TRUE)
    If(bGreaterThan)
        Message("First string is greater than second
            string", 5)
    Else
        Message("First string is not greater than second
            string", 5)
    End_If
Return
```

---

## See Also

[String\\_Empty](#), [String\\_Less\\_Than](#), [String\\_Less\\_Than\\_Or\\_Equal](#), [String\\_Equal](#),  
[String\\_Greater\\_Than](#), [String\\_Not\\_Equal](#), [String\\_Set](#), [Number\\_To\\_String\\_Decimal](#),  
[Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#), [Get\\_Scan\\_Type\\_Name](#)

---

## String\_Greater\_Than

Compare the two strings and determine whether one follows the other in alphabetical order.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Test1</i>          | Gets compared with Test2.  |
| <i>Test2</i>          | Gets compared with Test1.  |
| <i>Maximum Length</i> | Indicates whether the number of characters is limited.                 |
| <i>Ignore Case</i>    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

`String_Greater_Than ("Test1", "Test2", Maximum Length, Ignore Case)`

---

### Return Value

Returns a Boolean. TRUE if Test1 follows Test2 in alphabetical ordering, FALSE otherwise.

---

### Remarks

If the `Maximum Length` value is greater than 0, any characters after the specified number of characters are ignored. If `Ignore Case` is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script( String_Greater_Than_Test )
String( strEntered1 )
String( strEntered2 )
Boolean( bGreaterThan )
Activate( From_Menu )
    strEntered1 = Ask_String( "Type the first string",
"String_Greater_Than_Test", 0, 0, "" )
    strEntered2 = Ask_String( "Type the second string",
"String_Greater_Than_Test", 0, 0, "" )
    bGreaterThan = String_Greater_Than( strEntered1, strEntered2, 0, TRUE
)
    If( bGreaterThan )
        Message( "First string is greater than second string", 5 )
    Else
        Message( "First string is not greater than second string", 5 )
    End_If
Return
```

---

## See Also

[String\\_Empty](#), [String\\_Less\\_Than](#), [String\\_Less\\_Than\\_Or\\_Equal](#), [String\\_Equal](#),  
[String\\_Greater\\_Than\\_Or\\_Equal](#), [String\\_Not\\_Equal](#), [String\\_Set](#),  
[Number\\_To\\_String\\_Decimal](#), [Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#),  
[Get\\_Scan\\_Type\\_Name](#)

---

## String\_Not\_Equal

Compare the two strings and return TRUE if they do not have the same value.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Test1</i>          | Gets compared with Test2.  |
| <i>Test2</i>          | Gets compared with Test1.  |
| <i>Maximum Length</i> | Indicates whether the number of characters is limited.                 |
| <i>Ignore Case</i>    | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
String_Not_Equal ( "Test1", "Test2", Maximum Length, Ignore Case )
```

---

### Return Value

Returns a Boolean. FALSE if Test1 and Test2 are the same string, TRUE otherwise.

---

### Remarks

If the Maximum Length value is greater than 0, any characters after the specified number of characters are ignored. If Ignore Case is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script( String_Not_Equal_Test )  
String( strEntered1 )
```

```
String( strEntered2 )
Boolean( bNotEqual )
Activate( From_Menu )
    strEntered1 = Ask_String( "Type the first string",
"String_Not_Equal_Test", 0, 0, "" )
    strEntered2 = Ask_String( "Type the second string",
"String_Not_Equal_Test", 0, 0, "" )
    bNotEqual = String_Not_Equal( strEntered1, strEntered2, 0, TRUE )
    If( bNotEqual )
        Message( "First string is not equal to second string", 5 )
    Else
        Message( "First string is equal to second string", 5 )
    End_If
Return
```

---

### **See Also**

[String\\_Empty](#), [String\\_Less\\_Than](#), [String\\_Less\\_Than\\_Or\\_Equal](#), [String\\_Equal](#),  
[String\\_Greater\\_Than\\_Or\\_Equal](#), [String\\_Greater\\_Than](#), [String\\_Set](#),  
[Number\\_To\\_String\\_Decimal](#), [Ask\\_String](#), [Get\\_Screen\\_Text](#), [Speech\\_To\\_Text](#),  
[Get\\_Scan\\_Type\\_Name](#)

---

## Set\_Field\_Data\_ID

Sets the Data ID for a field.

---

### Parameters

|                       |   |
|-----------------------|---|
| <i>Field Index</i>    | The numeric index of the 5250 data field, index 0 is the first field.                           |
| <i>Data ID String</i> | The data identifier for the 5250 data field, blank to clear all data identifiers for the field. |

---

### Format

```
Set_Field_Data_ID (Field Index, "Data ID String")
```

---

### Return Value

Returns a Boolean. TRUE if successful, FALSE if the field index is not valid.

---

### Remarks

A field may have more than one Data ID. If the field already has a Data ID, this command will add another Data ID. Use a blank string to clear all Data IDs for the field.

Use actions like `Get_Field_Index()` to determine the index of a field.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

---

### Example

```
Script(Set_Field_Data_ID_Test)  
Boolean(bSetOK)
```

```
Activate(From_Menu)
  bSetOK = Set_Field_Data_ID(0, "N")
  If (bSetOK)
    Message("Set_Field_Data_ID worked", 5)
  Else
    Message("Set_Field_Data_ID failed", 5)
  End_If
Return
```

---

## See Also

[Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#),  
[Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#), [Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Set\_Field\_Symbology\_ID

Sets the Symbology ID for a field.

---

### Parameters

|                            |  |
|----------------------------|--|
| <i>Field Index</i>         | The numeric index of the 5250 data field, index 0 is the first field.        |
| <i>Symbology ID</i>        | The name of the symbology.   |
| <i>And-Or with Data ID</i> | Indicates whether the field data must match the Data ID and/or Symbology ID. |

---

### Format

`Set_Field_Symbology_ID (Field Index, "Symbology ID", And-Or with Data ID)`

---

### Return Value

If `And-Or with Data ID` is TRUE, then the field data must match both the Data ID and the Symbology ID. If `And-Or` is FALSE, then the field data must match either the Data ID or the Symbology ID.

---

### Remarks

A field may have more than one Symbology ID. If the field already has a Symbology ID, this command will add another Symbology ID. Use Symbology ID ANY to clear the symbologies, which will then allow you to use All Symbologies. ANY causes `Get_Num_Field_Symbology_IDs ()` to return zero, and `Get_Field_Symbology_ID ()` to return an empty string.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

Scanner symbology values can be found in *Symbologies and Values* on page 322.

---

### Example

```
Script( Set_Field_Symbology_ID_Test )
Boolean( bSetOK )
Activate( From_Menu )
    bSetOK = Set_Field_Symbology_ID( 2, "UPCA", FALSE )
    If( bSetOK )
        Message( "Set_Field_Symbology_ID worked", 5 )
    Else
        Message( "Set_Field_Symbology_ID failed", 5 )
    End_If
Return
```

---

### See Also

[Set\\_Field\\_Data\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#),  
[Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Get\_Field\_Symbology\_Operator

Query whether the field data must match both the field Data ID and the field Symbology ID.

---

### Parameters

*Field Index*                      The numeric index of the 5250 data field, index 0 is the first field.

---

### Format

Get\_Field\_Symbology\_Operator (Field Index)

---

### Return Value

Returns a Boolean. TRUE if the field data must match both the Data ID and the Symbology ID. Returns FALSE if the field data must match either the Data ID or Symbology ID.

---

### Remarks

Use the `Set_Field_Symbology_ID And-Or` parameter to set whether the field data must match both the field Data ID and the field Symbology ID.

---

### Example

```
Script( Get_Field_Symbology_Operator_Test )
Boolean( bSetOK )
Boolean( bOperator )
Activate( From_Menu )
    bSetOK = Set_Field_Data_ID( 0, "N" )
    bSetOK = Set_Field_Symbology_ID( 0, "UPCA", TRUE )
    bOperator = Get_Field_Symbology_Operator( 0 )
```

```
    If( bOperator )
        Message( "Field data must match both the Data ID and the Symbology
ID", 15 )
    Else
        Message( "Field data must match one or the other of Data ID and
Symbology ID", 15 )
    End_If
Return
```

---

## See Also

[Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#),  
[Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Set\_Field\_Append\_Scan\_Data

Controls whether to append scan data in the field.

---

### Parameters

|                         |  |
|-------------------------|--|
| <i>Field Index</i>      | The numeric index of the 5250 data field, index 0 is the first field.                      |
| <i>Append Scan Data</i> | Use TRUE to cause the scan data to be appended to the scanned value when scanning a field. |

---

### Format

Set\_Field\_Append\_Scan\_Data (Field Index, Append Scan Data)

---

### Return Value

Returns a Boolean. TRUE if successful, returns FALSE if the field index is not valid.

---

### Remarks

This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Set_Field_Append_Scan_Data_Test)
Boolean(bSetOK)
Activate(From_Menu)
  bSetOK = Set_Field_Append_Scan_Data(0, FALSE)
  If(bSetOK)
    Message("Set_Field_Append_Scan_Data worked", 5)
  Else
    Message("Set_Field_Append_Scan_Data failed", 5)
```

End\_If  
Return

---

## **See Also**

[Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#),  
[Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Set\_Field\_Com\_Data\_Field

Sets a field to be the Com Data Field for the screen.

---

### Parameters

|                           |   |
|---------------------------|---|
| <i>Field Index</i>        | The numeric index of the 5250 data field, index 0 is the first field. |
| <i>Set Com Data Field</i> | Use TRUE to make this field a Com Data Field.                         |

---

### Format

```
Set_Field_Com_Data_Field (Field Index, Set Com Data Field)
```

---

### Return Value

Returns a Boolean. TRUE if successful, FALSE if the index is not valid.

---

### Remarks

There can be only one Com Data Field per screen. Use FALSE to remove the Com Data Field setting.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Set_Field_Com_Data_Field_Test)
Boolean(bSetOK)
Activate(From_Menu)
  bSetOK = Set_Field_Com_Data_Field(2, FALSE)
```

```
If (bSetOK)
    Message("Set_Field_Com_Data_Field worked", 5)
Else
    Message("Set_Field_Com_Data_Field failed", 5)
End_If
Return
```

---

## See Also

[Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#),  
[Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#), [Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Set\_Field\_Prefix\_Scan\_Data

Sets the data prefixed to a field when the field is scanned.

---

### Parameters

|                       |   |
|-----------------------|---|
| <i>Field Index</i>    | The numeric index of the 5250 data field, index 0 is the first field. |
| <i>Data To Prefix</i> | Gets prefixed to the field data.                                      |

---

### Format

```
Set_Field_Prefix_Scan_Data (Field Index, "Data To Prefix")
```

---

### Return Value

Returns a Boolean. TRUE if successful, FALSE if the field index is not valid.

---

### Remarks

Use a blank string to clear the prefix data.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Set_Field_Prefix_Scan_Data_Test)
Boolean(bSetOK)
Activate(From_Menu)
  bSetOK = Set_Field_Prefix_Scan_Data(0, "99")
  If(bSetOK)
    Message("Set_Field_Prefix_Scan_Data worked", 5)
```

```
Else
    Message("Set_Field_Prefix_Scan_Data failed", 5)
End_If
Return
```

---

## See Also

[Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#),  
[Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#), [Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Get\_Field\_Append\_Scan\_Data

Query whether data is appended when the field is scanned.

---

### Parameters

*Field Index*                      The numeric index of the 5250 data field, index 0 is the first field.

---

### Format

Get\_Field\_Append\_Scan\_Data (Field Index)

---

### Remarks

This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Append_Scan_Data_Test)
Boolean(bAppend)
Activate(From_Menu)
  bAppend = Get_Field_Append_Scan_Data(0)
  If(bAppend)
    Message("Appending scan data", 5)
  Else
    Message("Not appending scan data", 5)
  End_If
Return
```

---

### See Also

[Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Data\\_ID](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),

[Get\\_Field\\_Com\\_Data\\_Field](#), [Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#),  
[Get\\_Field\\_Prefix\\_Scan\\_Data](#)

---

## Number\_Less\_Than

Compares two numbers and returns TRUE if Test1 is smaller than Test2, FALSE otherwise.

---

### Parameters

|              |                         |
|--------------|-------------------------|
| <i>Test1</i> | Gets compared to Test2. |
| <i>Test2</i> | Gets compared to Test1. |

---

### Format

```
Number_Less_Than (Test1, Test2)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 is smaller than Test2, FALSE otherwise.

---

### Remarks

The largest number is 2147483647. The smallest number is -2147483648

---

### Example

```
Script(Number_Less_Than_Test)
Boolean(bLessThan)
Number(nEntered1)
Number(nEntered2)
Activate(From_Menu)
    nEntered1 = Ask_Number("Type the first number",
        "Number_Less_Than_Test", 0, 2147483647, 0)
    nEntered2 = Ask_Number("Type the second number",
        "Number_Less_Than_Test", 0, 999, 0)
    bLessThan = Number_Less_Than(nEntered1, nEntered2)
```

```
If (bLessThan)
  Message("First number is less than second number", 5)
Else
  Message("First number is not less than second number",
  5)
End_If
Return
```

---

## **See Also**

[Number\\_Less\\_Than\\_Or\\_Equal](#), [Number\\_Equal](#), [Number\\_Greater\\_Than\\_Or\\_Equal](#),  
[Number\\_Greater\\_Than](#), [Number\\_Not\\_Equal](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

## Number\_Less\_Than\_Or\_Equal

Compare two numbers and determine if one is less than the other or if they are equal.

---

### Parameters

*Test1* Gets compared to Test2.

*Test2* Gets compared to Test1.

---

### Format

```
Number_Less_Than_Or_Equal (Test1, Test2)
```

---

### Return Value

Returns a Boolean. TRUE if Test1 is no greater than Test2, FALSE otherwise.

---

### Example

```
Script(Number_Less_Than_Or_Equal_Test)
Number(nEntered1)
Number(nEntered2)
Boolean(bLessThan)
Activate(From_Menu)
    nEntered1 = Ask_Number("Type the first number",
        "Number_Less_Than_Or_Equal_Test", 0, 999, 0)
    nEntered2 = Ask_Number("Type the second number",
        "Number_Less_Than_Or_Equal_Test", 0, 999, 0)
    bLessThan = Number_Less_Than_Or_Equal(nEntered1,
        nEntered2)
    If(bLessThan)
        Message("First number is less than or equal to the
            second number", 5)
    Else
        Message("First number is not less than or equal to the
            second number", 5)
```

End\_If  
Return

---

**See Also**

[Number\\_Less\\_Than](#), [Number\\_Equal](#), [Number\\_Greater\\_Than\\_Or\\_Equal](#),  
[Number\\_Greater\\_Than](#), [Number\\_Not\\_Equal](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

# Number\_Equal

Compare two numbers and determine whether they are equal.

---

## Parameters

*Test1* Gets compared to Test2.

*Test2* Gets compared to Test1.

---

## Format

Number\_Equal (Test1, Test2)

---

## Return Value

Returns a Boolean. TRUE if Test1 is the same as Test2, FALSE otherwise.

---

## Example

```
Script(Number_Equal_Test)
Number(nEntered1)
Number(nEntered2)
Boolean(bEqual)
Activate(From_Menu)
    nEntered1 = Ask_Number("Type the first number",
        "Number_Equal_Test", 0, 999, 0)
    nEntered2 = Ask_Number("Type the second number",
        "Number_Equal_Test", 0, 999, 0)
    bEqual = Number_Equal(nEntered1, nEntered2)
    If(bEqual)
        Message("First number is equal to the second number",
            5)
    Else
        Message("First number is not equal to the second
            number", 5)
```

End\_If  
Return

---

**See Also**

[Number\\_Less\\_Than](#), [Number\\_Less\\_Than\\_Or\\_Equal](#), [Number\\_Greater\\_Than\\_Or\\_Equal](#),  
[Number\\_Greater\\_Than](#), [Number\\_Not\\_Equal](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

## Number\_Greater\_Than\_Or\_Equal

Compare two numbers and determine if one is greater than the other or if they are equal.

---

### Parameters

*Test1* Gets compared to Test2.

*Test2* Gets compared to Test1.

---

### Format

Number\_Great\_Than\_Or\_Equal (Test1, Test2)

---

### Return Value

Returns a Boolean. TRUE if Test1 is no smaller than Test2, FALSE otherwise.

---

### Example

```
Script(Number_Greater_Than_Or_Equal_Test)
Number(nEntered1)
Number(nEntered2)
Boolean(bGreaterThan)
Activate(From_Menu)
    nEntered1 = Ask_Number("Type the first number",
        "Number_Greater_Than_Or_Equal_Test", 0, 999, 0)
    nEntered2 = Ask_Number("Type the second number",
        "Number_Greater_Than_Or_Equal_Test", 0, 999, 0)
    bGreaterThan = Number_Greater_Than_Or_Equal(nEntered1,
        nEntered2)
    If(bGreaterThan)
        Message("First number is greater than or equal to the
            second number", 5)
    Else
        Message("First number is not greater than or equal to
            the second number", 5)
```

End\_If  
Return

---

**See Also**

[Number\\_Less\\_Than](#), [Number\\_Less\\_Than\\_Or\\_Equal](#), [Number\\_Equal](#),  
[Number\\_Greater\\_Than](#), [Number\\_Not\\_Equal](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

# Number\_Greater\_Than

Compare two numbers and determine if one is greater than the other.

---

## Parameters

*Test1* Gets compared to Test2.

*Test2* Gets compared to Test1.

---

## Format

Number\_Greater\_Than (Test1, Test2)

---

## Return Value

Returns a Boolean. TRUE if Test1 is larger than Test2, FALSE otherwise.

---

## Example

```
Script(Number_Greater_Than_Test)
Number(nEntered1)
Number(nEntered2)
Boolean(bGreaterThan)
Activate(From_Menu)
    nEntered1 = Ask_Number("Type the first number",
        "Number_Greater_Than_Test", 0, 999, 0)
    nEntered2 = Ask_Number("Type the second number",
        "Number_Greater_Than_Test", 0, 999, 0)
    bGreaterThan = Number_Greater_Than(nEntered1,
        nEntered2)
    If(bGreaterThan)
        Message("First number is greater than second number",
            5)
    Else
        Message("First number is not greater than second
            number", 5)
```

End\_If  
Return

---

### **See Also**

[Number\\_Less\\_Than](#), [Number\\_Less\\_Than\\_Or\\_Equal](#), [Number\\_Equal](#),  
[Number\\_Greater\\_Than\\_Or\\_Equal](#), [Number\\_Not\\_Equal](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

# Number\_Not\_Equal

Compares two numbers.

---

## Parameters

*Parameter1 (Number) : "Test1"* Gets compared to Test2

*Parameter1 (Number) : "Test2"* Gets compared to Test1.

---

## Format

Number\_Not\_Equal:"Test1"

---

## Return Value

Returns a Boolean. FALSE if Test1 is the same as Test2, TRUE otherwise.

---

## Example

```
Script( Number_Not_Equal_Test )
Boolean( bNotEqual )
Number( nEntered1 )
Number( nEntered2 )
Activate( From_Menu )
    nEntered1 = Ask_Number( "Type the first number",
"Number_Not_Equal_Test", 0, 999, 15 )
    nEntered2 = Ask_Number( "Type the second number",
"Number_Not_Equal_Test", 0, 999, 704 )
    bNotEqual = Number_Not_Equal( nEntered1, nEntered2 )
    If( bNotEqual )
        Message( "First number is not equal to the second number", 8 )
    Else
```

```
        Message( "First number is equal to the second number", 8 )  
    End_If
```

---

## **See Also**

[Number\\_Less\\_Than](#), [Number\\_Less\\_Than\\_Or\\_Equal](#), [Number\\_Equal](#),  
[Number\\_Greater\\_Than\\_Or\\_Equal](#), [Number\\_Greater\\_Than](#), [Number\\_Set](#), [Ask\\_Number](#),  
[String\\_To\\_Number\\_Decimal](#)

---

## Ask\_OK\_Cancel

Displays a message in a dialog box with an **OK** and **Cancel** button and waits until the user selects a button.

---

### Parameters

|                            |   |
|----------------------------|---|
| <i>Message Text</i>        | The message that the script displays in the message box.                  |
| <i>Title Text</i>          | The message that the script displays in the title bar of the message box. |
| <i>Make Cancel Default</i> | Indicates whether the <b>Cancel</b> button is the default button.         |

---

### Format

```
Ask_OK_Cancel ("Message Text", "Title Text", Make Cancel Default)
```

---

### Return Value

Returns a Boolean. TRUE if the user selects **OK**, returns FALSE if the user selects **Cancel**.

---

### Example

```
Script(Ask_OK_Cancel_Test)
Boolean(bResult)
Activate(From_Menu)
  bResult = Ask_OK_Cancel("OK for TRUE, Cancel for
  FALSE", "Ask_OK_Cancel_Test", FALSE)
  If(bResult)
    Message("Selected: OK", 5)
  Else
    Message("Selected: Cancel", 5)
  End_If
```

Return

---

**See Also**

[Ask\\_Yes\\_No](#), [Ask\\_Yes\\_No\\_Cancel](#), [Ask\\_Ok](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [Message](#), [Ask\\_Number](#)

---

## Ask\_Yes\_No

Displays a message in a box with a **Yes** and **No** button and waits until the user selects a button.

---

### Parameters

|                            |   |
|----------------------------|---|
| <i>Message Text</i>        | The message that the script displays in the message box.                  |
| <i>Title Text</i>          | The message that the script displays in the title bar of the message box. |
| <i>Make Cancel Default</i> | Indicates whether the <b>No</b> button is the default button.             |

---

### Format

```
Ask_Yes_No ("Message Text", "Title Text", Make Cancel Default)
```

---

### Return Value

Returns a Boolean. TRUE if the user selects **Yes**, returns FALSE if the user selects **No**.

---

### Example

```
Script(Ask_Yes_No_Test)
Boolean(bResult)
Activate(From_Menu)
  bResult = Ask_Yes_No("Yes for TRUE, No for FALSE",
  "Ask_Yes_No_Test", FALSE)
  If(bResult)
    Message("Selected: Yes", 5)
  Else
    Message("Selected: No", 5)
  End_If
```

Return

---

**See Also**

[Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No\\_Cancel](#), [Ask\\_Ok](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [Message](#), [Ask\\_Number](#)

---

# Suspend

Suspends the device.

---

## Parameters

|                           |  |
|---------------------------|--|
| <i>Prefer Hibernation</i> | Indicates whether the device will hibernate instead of suspending. |
| <i>Force Suspension</i>   | Indicates whether other applications can override the suspension.  |

---

## Format

Suspend (Prefer Hibernation, Force Suspension)

---

## Return Value

Returns a Boolean. TRUE if the device was suspended; returns FALSE otherwise.

---

## Remarks

If *Prefer Hibernation* is TRUE, the device will hibernate instead of suspend. (Not supported on CE devices.)

If *Force Suspension* is TRUE, the device will suspend immediately and other applications will not be allowed to override.

---

## Example

```
Script(Suspend_Test)
Boolean(bHibernate)
Boolean(bForce)
Boolean(bResult)
```

```
Activate(From_Menu)
    bHibernate = Ask_Yes_No("Hibernate instead of
    Suspend?", "Suspend_Test", FALSE)
    bForce = Ask_Yes_No("Force Suspend?", "Suspend_Test",
    FALSE)
    bResult = Suspend(bHibernate, bForce)
    Return
```

---

## **See Also**

[Reboot](#), [Get\\_Time\\_Since\\_Reset](#), [Ask\\_Yes\\_No](#), [Disconnect](#), [Abort\\_All](#), [Exit\\_Application](#)

---

## Search\_Screen

Searches the screen for the supplied text.

---

### Parameters

|                          |  |
|--------------------------|--|
| <i>Search String</i>     | The string to find on the screen.                                      |
| <i>Top Search Row</i>    | The row in which the search begins.                                    |
| <i>Bottom Search Row</i> | The row in which the search ends.                                      |
| <i>Ignore Case</i>       | Indicates whether the case of the letters is taken into consideration. |

---

### Format

`Search_Screen (Search String, Top Search Row, Bottom Search Row, Ignore Case)`

---

### Return Value

Returns a Boolean. TRUE if the text is found, FALSE otherwise.

---

### Remarks

The rows to be searched can be specified, where 1 is the top row. If the bottom row value is less than 1, searching continues to the bottom of the screen. If `Ignore Case` is TRUE, then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script( Search_Screen_Test )  
String( strSearch )  
Boolean( bFound )
```

```
Boolean( bIgnoreCase )
Number( nStartRow )
Number( nEndRow )
Activate( From_Menu )
    strSearch = Ask_String( "Enter string to search for",
"Search_String_Test", 0, 999, "" )
    nStartRow = Ask_Number( "Enter starting row (1 is top of
screen)", "Search_String_Test", 0, 999, 0 )
    bIgnoreCase = Ask_Yes_No( "Ignore case?", "Search_String_Test", FALSE
)
    bFound = Search_Screen( strSearch, nStartRow, nEndRow, bIgnoreCase )
    If( bFound )
        Message( "Search string found.", 5 )
    Else
        Message( "Search string not found.", 5 )
    End_If
Return
```

---

## See Also

[Get\\_Screen\\_Rows](#), [Get\\_Position\\_Row](#), [Get\\_Field\\_Row](#), [Get\\_Field\\_Index](#),  
[Set\\_Field\\_Data\\_ID](#), [String\\_Set](#), [Ask\\_String](#)

---

# Web\_Navigate

Navigates WEB emulation to the URL provided.

---

## Parameters

|            |  |
|------------|--|
| <i>URL</i> | The URL to which the Web emulator navigates. |
|------------|--|

---

## Format

```
Web_Navigate ("URL")
```

---

## Return Value

Returns a Boolean. TRUE if the navigation was successful, returns FALSE otherwise.

---

## Example

```
Script(Web_Navigate_to_Google)
Activate(From_Menu)
  Comment: This script when launched will navigate to www.google.com
  Web_Navigate("http://www.google.com")
  Return
```

---

## See Also

[Web\\_Navigate\\_Frame](#), [Web\\_Navigate\\_Post\\_Data](#), [Web\\_Scripting](#), [Web\\_Search\\_Source](#), [Web\\_Get\\_Current\\_Element](#), [Web\\_Get\\_Source](#), [String\\_Set](#)

## Web\_Navigate\_Frame

Navigates WEB emulation to the URL provided within the indicated frame (`\Frame Name\`).

---

### Parameters

|                   |   |
|-------------------|---|
| <i>URL</i>        | The URL to which the Web emulator navigates.            |
| <i>Frame Name</i> | The name of the frame where the navigation takes place. |

---

### Format

```
Web_Navigate_Frame ("URL", "Frame Name")
```

---

### Return Value

Returns a Boolean. TRUE if the navigation was successful, returns FALSE otherwise.

---

### Example

```
Script(Web_Navigate_Frame_to_Google)
Activate(From_Menu)
  Comment: This script when launched will navigate to
  www.google.com.
  Web_Navigate_Frame("http://www.google.com",
  "FrameName")
Return
```

---

### See Also

Web\_Navigate, Web\_Navigate\_Post\_Data, Web\_Scripting, Web\_Search\_Source, ,  
Web\_Get\_Current\_Element, Web\_Get\_Source, String\_Set

## Web\_Navigate\_Post\_Data

Navigates WEB emulation to the URL provided.

---

### Parameters

|                  |   |
|------------------|---|
| <i>URL</i>       | The URL to which the Web emulator navigates.                  |
| <i>Post Data</i> | The string of data that the Web emulator sends to the server. |

---

### Format

```
Web_Navigate_Post_Data ("URL", "Post Data")
```

---

### Return Value

Returns a Boolean. TRUE if the navigation was successful, returns FALSE otherwise.

---

### Remarks

The `\Post Data\` is sent to the server using an HTTP POST transaction rather than an HTTP GET transaction.

---

### Example

```
Script( Web_Navigate_Post_Data_Test )  
Activate( From_Menu )
```

Comment: This script when launched will post a first and last name to <http://www.snee.com/xml/crud/posttest.cgi>

```
Web_Navigate_Post_Data( "http://www.snee.com/xml/crud/posttest.cgi",  
"fname=MyFirstName&lname=MyLastName" )  
Return
```

---

**See Also**

[Web\\_Navigate](#), [Web\\_Navigate\\_Frame](#), [Web\\_Scripting](#), [Web\\_Search\\_Source](#), ,  
[Web\\_Get\\_Current\\_Element](#), [Web\\_Get\\_Source](#), [String\\_Set](#)

## Web\_Scripting

Instructs WEB emulation to execute the scripting information.

---

### Parameters

|             |  |
|-------------|--|
| <i>Code</i> | The JavaScript or VBScript that the Web emulator will execute. |
|-------------|--|

---

### Format

```
Web_Scripting ("Code")
```

---

### Return Value

Returns a Boolean. TRUE if the script execution started successfully, returns FALSE otherwise.

---

### Remarks

The code should start with a `\javascript:\` or `\vbscript:\` string to ensure that the correct scripting type is used.

---

### Example

```
Script(Web_Scripting_Test)
Activate(From_Menu)
    Web_Scripting("javascript:alert('This message shows
    that WLTN scripting executed a javascript alert
```

```
command' ) ")  
Return
```

---

**See Also**

[Web\\_Navigate](#), [Web\\_Navigate\\_Frame](#), [Web\\_Navigate\\_Post\\_Data](#), [Web\\_Search\\_Source](#), ,  
[Web\\_Get\\_Current\\_Element](#), [Web\\_Get\\_Source](#), [String\\_Set](#)

## Web\_Search\_Source

Searches the page source of the current WEB emulation page.

---

### Parameters

|                      |  |
|----------------------|--|
| <i>Search Text</i>   | The text to search for in the page source.                             |
| <i>Ignore Case</i>   | Indicates whether the case of the letters is taken into consideration. |
| <i>Search Frames</i> | Indicates whether the page source for the frames is searched as well.  |

---

### Format

```
Web_Search_Source ("Search Text", Ignore Case, Search Frames)
```

---

### Return Value

Returns a Boolean. TRUE if the text is found anywhere in the page source, returns FALSE otherwise.

---

### Remarks

If `\Search Frames\` is TRUE, the page source of any frames will be searched as well.

---

### Example

```
Script(Web_Search_Source_Test)
Boolean(bFound)
Activate(From_Menu)
    bFound = Web_Search_Source("wavelink", TRUE, TRUE)
    If (bFound)
        Message("I found the word WAVELINK", 5)
```

End\_If  
Return

---

**See Also**

[Web\\_Navigate](#), [Web\\_Navigate\\_Frame](#), [Web\\_Navigate\\_Post\\_Data](#), [Web\\_Scripting](#), ,  
[Web\\_Get\\_Current\\_Element](#), [Web\\_Get\\_Source](#), [String\\_Set](#)

## Speech\_From\_Text\_Available

Determines whether text-to-speech is supported.

---

### Return Value

Returns a Boolean. TRUE if text-to-speech is supported on the computer, returns FALSE otherwise.

---

### Example

```
Script(Speech_From_Text_Available_Test)
Boolean(bAvailable)
Activate(From_Menu)
    bAvailable = Speech_From_Text_Available
    If(bAvailable)
        Message("Speech From Text is available", 5)
    Else
        Message("Speech From Text is not available", 5)
    End_If
Return
```

---

### See Also

[Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#),  
[Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_From\_Text

Converts text into sound and plays the resulting sound on the computer.

---

### Parameters

|                        |   |
|------------------------|---|
| <i>Text</i>            | The text that is converted into sound.  |
| <i>Wait Until Done</i> | Indicates whether the script resumes execution before the speech has completely played. |

---

### Format

```
Speech_From_Text ("Text", Wait Until Done)
```

---

### Return Value

Returns a Boolean. TRUE if the sound was played successfully, FALSE if otherwise.

---

### Remarks

If `\Wait Until Done\` is FALSE, the script will continue to execute while the sound is being played.

---

### Example

```
Script(Speech_From_Text_Test)  
Activate(From_Menu)
```

```
Speech_From_Text("Hello again.", TRUE)
Return
```

---

## See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#),  
[Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

---

## Speech\_To\_Text\_Available

Determines whether speech-to-text is supported.

---

### Return Value

Returns a Boolean. TRUE if speech-to-text is supported on the computer, returns FALSE otherwise.

---

### Example

```
Script(Speech_To_Text_Available_Test)
Boolean(bAvailable)
Activate(From_Menu)
    bAvailable = Speech_To_Text_Available
    If(bAvailable)
        Message("Speech To Text is available", 5)
    Else
        Message("Speech To Text is not available", 5)
    End_If
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text](#),  
[Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_Setting\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_To\_Text

Listens to the user speak and returns the text equivalent of what he/she said in the string variable.

---

### Parameters

|                |  |
|----------------|--|
| <i>Text</i>    | The string variable into which the converted speech is placed. |
| <i>Grammar</i> | The grammar file to use.                                       |

---

### Return Value

Returns a string, the text equivalent of a user's speech. Returns an empty string if no acceptable speech was detected.

---

### Remarks

If a grammar is specified, the grammar file with that name is used for speech recognition; otherwise, the previous grammar file is reused.

---

### Example

```
Script( Speech_To_Text_Test )
String( szResult )
Activate( From_Menu )
  If_Not( Speech_To_Text_Available )
    Ask_OK( "Speech to text is not available.", "Error" )
    Return
  End_If
  Message( "Say one or more digits", 0 )
  If_Not( Speech_To_Text( szResult, "connected_digits" ) )
    Message_Clear
    Ask_OK( "No results returned from the speech.", "Error" )
    Return
  End_If
  Message_Clear
```

```
Ask_OK( szResult, "Speech-to-text-results" )  
Return
```

---

## See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_To\_Text\_No\_Wait

Listens to the user speak and returns the text equivalent of what the user said.

---

### Parameters

|                |  |
|----------------|--|
| <i>Done</i>    | Indicates when the speech is recognized or times out.          |
| <i>Text</i>    | The string variable into which the converted speech is placed. |
| <i>Grammar</i> | The grammar file to use.                                       |

---

### Format

`Speech_To_Text_No_Wait (Done, Text, Grammar)`

---

### Return Value

Returns the text equivalent of a user's speech in the string variable.

---

### Remarks

The boolean variable is set to TRUE when the speech is recognized or times out. If a grammar is specified, the grammar file with that name is used for the speech recognition. Otherwise, the previous grammar file is reused.

---

### Example

```
Script( Speech_Demo )
Profile( Speech_Demo_Profile )
String( sSpeechResult )
Boolean( bSpeechStarted )
```

```
Boolean( bSpeechDone )
Number( nCurrentGrammar )
Number( nDesiredGrammar )
Number( nCurrentScreen )
Activate( Connection )
    Comment: Is Speech available?
    If_Not( Speech_To_Text_Available )
        Message( "Speech To Text Not available", 3 )
        Return
    End_If
    If_Not( Speech_From_Text_Available )
        Message( "Speech From Text Not Available", 3 )
        Return
    End_If

    Comment: We are using English.
    Speech_Change_Setting( "stt_language_long",
        Speech_Find_Setting_Value( "stt_language_long", "English", FALSE ) )
    Speech_Change_Setting( "tts_language_long",
        Speech_Find_Setting_Value( "tts_language_long", "English", FALSE ) )

While( TRUE )

    Comment: Perform Speech-to-Text with the desired grammar.
    If( Number_Not_Equal( nCurrentGrammar, nDesiredGrammar ) )
        Speech_To_Text_Cancel
        bSpeechStarted = FALSE
        bSpeechDone = FALSE
        sSpeechResult = ""
    End_If
    If( Boolean_And( bSpeechDone, String_Empty( sSpeechResult ) ) )
        Comment: The string was cleared because the result was used.
        Reset for the next use.
        bSpeechStarted = FALSE
        bSpeechDone = FALSE
    End_If
    If_Not( bSpeechStarted )
        If( Number_Equal( nDesiredGrammar, 1 ) )
            Comment: Set the threshold for this grammar.
            Comment: Lower values are more likely to get results, but
                they are more likely to be wrong.
            Speech_Change_Setting( "stt_threshold", 4500 )

            Comment: Use the digit.bnf grammar file.
            bSpeechStarted = Speech_To_Text_No_Wait( bSpeechDone,
                sSpeechResult, "digit" )
```

```

End_If

Comment: Can add support for other grammars here.

nCurrentGrammar = nDesiredGrammar
End_If

Comment: Look for screens where we include speech support.
If( Boolean_And( String_Equal( Get_Screen_Text_Length( 1, 1, 6 ),
"Pick ", 0, FALSE ), String_Equal( Get_Screen_Text_Length( 7, 1,
4 ), "Pick", 0, FALSE ) ) )
    Comment: The first time we see this screen, tell the user what
    we want.
    If( Number_Not_Equal( nCurrentScreen, 101 ) )
        nCurrentScreen = 101
        Speech_From_Text( "Pick a menu item from 1 to 5.", FALSE )
    End_If

    Comment: Prepare to get the user's response.
    nDesiredGrammar = 1
    If( Number_Not_Equal( nDesiredGrammar, nCurrentGrammar ) )
        Continue
    End_If

    Comment: Handle any user responses.
    If_Not( String_Empty( sSpeechResult ) )
        If( Boolean_And( Number_Greater_Than_Or_Equal(
String_To_Number_Decimal( sSpeechResult ), 1 ),
Number_Less_Than_Or_Equal( String_To_Number_Decimal(
sSpeechResult ), 5 ) ) )
            Comment: Type the response the user supplied.
            Keypress_String( sSpeechResult )
            Keypress_Key( "VT220", "Enter" )
            Wait_For_Screen_Update
            nDesiredGrammar = 0
            sSpeechResult = ""
            Continue
        End_If

        Speech_From_Text( "Unexpected result. Please try again.",
FALSE )
        sSpeechResult = ""
        Continue
    End_If

    Comment: Wait for the user to respond.

```

```
        Wait_For_Screen_Update
        Continue
    End_If

    Comment: Can add support for other screens here.

    Comment: If we reach this point, we don't recognize the current
        screen.
    Comment: Wait for a screen we recognize.
    nCurrentScreen = 0
    nDesiredGrammar = 0
    Wait_For_Screen_Update
End_While

Return
```

---

## See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_To\_Text\_Cancel

Provides a way for the script to perform other functions while the speech-to-text action occurs.

---

### Return Value

Returns after cancelling the last `Speech_To_Text_No_Wait` action. Returns immediately if there is no action to cancel.

---

### Example

See the example for `Speech_To_Text_No_Wait`.

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#), [Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

---

## Speech\_Setting\_Available

Identifies speech settings by case-insensitive name strings.

---

### Parameters

*Setting* Check if this setting name is supported.

---

### Format

Speech\_Setting\_Available (Setting)

---

### Return Value

Returns a Boolean. TRUE if the speech setting mane is supported, FALSE otherwise.

---

### Remarks

Refer to the *Wavelink Voice-Enabled Emulation User Guide* for a list of available setting names.

---

### Example

```
Script(Speech_Setting_Available_Test)
Boolean(bAvailable)
  bAvailable = Speech_Setting_Available("stt_language")
  If(bAvailable)
    Message("Speech setting stt_language is available",
    5)
  Else
    Message("Speech setting stt_language is not
    available", 5)
  End_If
```

Return

---

### **See Also**

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

---

# Speech\_Change\_Setting

Changes the speech setting to the specified value.

---

## Parameters

|                |                                    |
|----------------|------------------------------------|
| <i>Setting</i> | The name of the setting to change. |
| <i>Value</i>   | The new value for the setting.     |

---

## Format

```
Speech_Change_Setting ("Setting", Value)
```

---

## Return Value

Returns a Boolean. TRUE if the setting is supported and the value is valid for that setting, returns FALSE otherwise.

---

## Example

```
Script(Speech_Change_Setting_Test)
  Comment: Increase the speech-to-text timeout to twenty seconds
  Speech_Change_Setting("stt_timeout", 20000)
  Return
```

---

## See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#), [Speech\\_Setting\\_Available](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#), [Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)



Speech\_To\_Text\_Get\_User\_Name, Speech\_To\_Text\_Change\_User\_Name  
Speech\_From\_Text\_Error\_Desc, Speech\_To\_Text\_Error\_Desc, Speech\_From\_Text\_Cancel

## Speech\_Get\_Setting\_Max

Gets the largest value for a speech setting.

---

### Parameters

|                |  |
|----------------|--|
| <i>Setting</i> | Get the maximum value for this setting name. |
|----------------|--|

---

### Format

```
Speech_Get_Setting_Max ("Setting")
```

---

### Return Value

Returns the largest possible value for a speech setting. Returns 0 if only one setting value is supported, returns -1 if the speech setting is not valid.

---

### Example

```
Script(Speech_Get_Setting_Max_Test)
String(strMessage)
Number(nTimeout)
Activate(From_Menu)
    nTimeout = Speech_Get_Setting_Max("stt_timeout")
    strMessage = String_Combine("stt_timeout maximum:",
    Number_To_String_Decimal(nTimeout))
    Message(strMessage, 5)
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),

Speech\_Setting\_Available, Speech\_Change\_Setting, Speech\_Get\_Setting,  
Speech\_Find\_Setting\_Value, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_To\_Text\_Get\_User\_Name, Speech\_To\_Text\_Change\_User\_Name  
Speech\_From\_Text\_Error\_Desc, Speech\_To\_Text\_Error\_Desc, Speech\_From\_Text\_Cancel

## Speech\_Find\_Setting\_Value

Searches all possible value descriptions for the speech setting.

---

### Parameters

|                          |  |
|--------------------------|--|
| <i>Setting</i>           | The name of the setting to search.                 |
| <i>Value Description</i> | The value description to match.                    |
| <i>Exact Only</i>        | Indicates whether only an exact match is returned. |

---

### Format

```
Speech_Find_Setting_Value ("Setting", Value Description, Exactly Only)
```

---

### Return Value

Returns the value of the setting that is the closest match. If `\“Exact Only\“` is TRUE, then only exact matches are returned. Returns `-1` if no match is found.

---

### Example

```
Script(Speech_Find_Setting_Value_Test)
String(strMessage)
Number(nLanguage)
Activate(From_Menu)
    nLanguage = Speech_Find_Setting_Value("stt_language",
    "enu", FALSE)
    strMessage = String_Combine("stt_language match for
    enu:", Number_To_String_Decimal(nLanguage))
```

```
Message(strMessage, 5)  
Return
```

---

## See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#)  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_Get\_Setting\_Value\_Desc

Get a description of the speech setting value.

---

### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>Setting</i> | The name of the setting.        |
| <i>Value</i>   | The description for this value. |

---

### Format

```
Speech_Get_Setting_Value_Desc ("Setting", Value)
```

---

### Return Value

Returns a string that describes the value for the speech setting (this does not need to be the setting's current value). Returns an empty string if the setting or value is not valid.

---

### Example

```
Script( Speech_Get_Setting_Value_Desc_Test )
String( strDescription )
Activate( From_Menu )
    strDescription = Speech_Get_Setting_Value_Desc( "stt_language", 1 )
    Message( strDescription, 7 )
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),

Speech\_To\_Text\_Get\_User\_Name, Speech\_To\_Text\_Change\_User\_Name  
Speech\_From\_Text\_Error\_Desc, Speech\_To\_Text\_Error\_Desc, Speech\_From\_Text\_Cancel

## Speech\_To\_Text\_Get\_User\_Name

Gets the user name.

---

### Return Value

Returns a string with the user name being used by the speech-to-text engine. An empty string is returned if no user name has been assigned.

---

### Example

```
Script( Speech_To_Text_Get_User_Name_Test )
String( strUsername )
Activate( From_Menu )
    strUsername = Speech_To_Text_Get_User_Name
    Message( strUsername, 4 )
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Change\\_User\\_Name](#), [Speech\\_From\\_Text\\_Error\\_Desc](#),  
[Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

---

## Speech\_To\_Text\_Change\_User\_Name

Changes the user name being used by the speech-to-text engine.

---

### Return Value

Returns a Boolean. TRUE if the user name was changed, FALSE if this feature is not supported.

---

### Example

```
Script( Speech_To_Text_Change_User_Name_Test )
String( strUsername )
Activate( From_Menu )
    strUsername = Ask_String( "Enter the new user name", "New User Name",
1,25, "" )
    Speech_To_Text_Change_User_Name( strUsername )
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_To\\_Text\\_Cancel](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_From\\_Text\\_Error\\_Desc](#),  
[Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_From\_Text\_Error\_Desc

Gets an error description for the last speech-from-text action.

---

### Return Value

Returns a string describing the last error from a speech-from-text action.

---

### Remarks

An empty string is returned if no errors have occurred.

---

### Example

```
Script( Speech_From_Text_Error_Desc_Test )
String( strDescription )
Activate( From_Menu )
    strDescription = Speech_From_Text_Error_Desc
    Message( String_Combine( "Last speech error:", strDescription ), 5 )
Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_To\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

---

## Speech\_To\_Text\_Error\_Desc

Gets an error description for the last speech-to-text action.

---

### Return Value

Returns a string describing the last error from a speech-to-text action.

---

### Remarks

An empty string is returned if no errors have occurred.

---

### Example

```
Script( Speech_To_Text_Error_Desc_Test )
String( strDescription )
Activate( From_Menu )
    strDescription = Speech_To_Text_Error_Desc
    Message( String_Combine( "Last speech-to-text error:", strDescription
),5 )
    Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#),  
[Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_Setting\\_Available](#),  
[Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#),  
[Speech\\_From\\_Text\\_Error\\_Desc](#), [Speech\\_From\\_Text\\_Cancel](#)

## Speech\_From\_Text\_Cancel

Provides a way for the script to perform other functions while the text-to-speech action occurs.

---

### Return Value

Returns after canceling any unspoken speech-from-text actions.

---

### Remarks

Returns immediately if there is no action to cancel.

---

### Example

```
Script(Speech_From_Text_Cancel_Test)
Activate(From_Menu)
  Speech_From_Text("The quick brown fox jumped over the
  lazy dogs.", FALSE)
  Delay(1000)
  Speech_From_Text_Cancel
  Return
```

---

### See Also

[Speech\\_From\\_Text\\_Available](#), [Speech\\_From\\_Text](#), [Speech\\_To\\_Text\\_Available](#), [Speech\\_To\\_Text](#), [Speech\\_To\\_Text\\_No\\_Wait](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_To\\_Text\\_Get\\_User\\_Name](#), [Speech\\_To\\_Text\\_Change\\_User\\_Name](#), [Speech\\_To\\_Text\\_Cancel](#)

## Wait\_For\_Screen\_Update\_With\_Timeout

Suspends the current script until the screen has been updated or the specified number of milliseconds have passed.

---

### Parameters

*Value1*            The number of milliseconds that the script suspends.

---

### Format

Wait\_For\_Screen\_Update\_With\_Timeout(Value1)

---

### Remarks

Returns TRUE if the screen was updated, FALSE if the timeout occurred.

---

### Example

```
Script(Wait_For_Screen_Update_With_Timeout_Test)
  Activate(From_Menu)
  Message("Waiting a few seconds for screen update", 0)
  Wait_For_Screen_Update_With_Timeout(3000)
  Message_Clear
  Return
```

# Keypress\_Capture\_Stop

Stops capturing the specified key and modifier combination.

---

## Parameters

|               |  |
|---------------|--|
| <i>Value1</i> | The Key Value (integer).               |
| <i>Value2</i> | The modifier (Shift, Ctrl, Alt, None). |

---

## Format

```
Keypress_Capture_Stop (Value1, "Value2")
```

---

## Remarks

Supported modifiers are Shift, Ctrl, Alt and None. An empty string is treated as None."

---

## Example

```
Script( Ctrl-F10_Keypress_Test )
Boolean( bKeyPressed )
Boolean( bF1Pressed )
Activate( From_Menu )
    Message( "F1 is captured; press control-F10 to exit", 0 )
    Keypress_Capture( 0x79, "Ctrl", bKeyPressed )
    Keypress_Capture( 0x70, "", bF1Pressed )
    While_Not( bKeyPressed )
        Wait_For_Screen_Update
        If( bF1Pressed )
            bF1Pressed = FALSE
            Keypress_Capture_Stop( 0x70, "" )
            Message( "F1 Pressed; capturing stopped for F1.", 3 )
        End_If
    End_While
Keypress_Capture_Stop( 0x79, "C" )
Keypress_Capture_Stop_All
```

```
Message( "Ctrl-F10 Pressed - script is done", 5 )  
Return
```

## Cancel\_Other\_Scripts

Cancels all other scripts for the session with the script name.

---

### Parameters

*Value1* This is the script name (a string).

---

### Format

```
Cancel_Other_Scripts("Value1")
```

---

### Remarks

If the name is left blank, all other scripts for the session are cancelled.

The calling script is never cancelled.

---

### Example

```
Script( Cancel_Script_All )
String( sMessage )
Number( nScriptsCancelled )
Activate( From_Menu )
    nScriptsCancelled = Cancel_Other_Scripts( "" )
    sMessage = String_Combine( Number_To_String_Decimal(
nScriptsCancelled ), " script(s) cancelled." )
    Ask_OK( sMessage, "Results" )
Return
```

## Printer\_Data

Sends data directly to the currently defined printer.

---

### Parameters

|               |                          |
|---------------|--------------------------|
| <i>Value1</i> | Print data (a string).   |
| <i>Value2</i> | All Data Sent (Boolean). |

---

### Format

```
Print_Data("Value1"), Value2
```

---

### Remarks

If "All Data Sent" is TRUE, then printing will begin after the latest data is sent. Otherwise, printing will wait for additional data.

---

### Example

```
Script( Printing_Example )
String( sScanData )
Number( sScanType )
Activate( On_Input, sScanData, sScanType )
    Comment: Prints a label containing displaying the barcode scanned.
    If_Not( Printer_Data( "! 0 200 200 148 1\0D\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "LABEL\0D\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "BARCODE CODE39 2 1 45 30 70 ", FALSE ) )
        Goto: Print_Error
    End_If
    If( Number_Greater_Than( String_Length( sScanData ), 20 ) )
        Printer_Cancel
```

```
        Message( "Barcode is too long.", 5 )
        Return
    End_If
    If_Not( Printer_Data( sScanData, FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "\0D\0A", TRUE ) )
        Goto: Print_Error
    End_If

    Comment: Offer to let the user print additional labels.
    While( Ask_Yes_No( "Do you want to print the label again?", "Reprint",
TRUE ) )
        If_Not( Printer_Repeat )
            Goto: Print_Error
        End_If
    End_While

    Message( "Printing successful.", 5 )
    Return

Label: Print_Error
    Printer_Cancel
    Message( "Unable to print the label.", 5 )
    Return
```

# Printer\_Repeat

Instructs the printer to reprint the last item printed.

---

## Remarks

If TRUE, the printer will produce the last item printed.

---

## Example

```
Script( Printing_Example )
String( sScanData )
Number( sScanType )
Activate( On_Input, sScanData, sScanType )
    Comment: Prints a label containing displaying the barcode scanned.
    If_Not( Printer_Data( "! 0 200 200 148 1\0D\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "LABEL\0D\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "BARCODE CODE39 2 1 45 30 70 ", FALSE ) )
        Goto: Print_Error
    End_If
    If( Number_Greater_Than( String_Length( sScanData ), 20 ) )
        Printer_Cancel
        Message( "Barcode is too long.", 5 )
        Return
    End_If
    If_Not( Printer_Data( sScanData, FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "\0D\0A", TRUE ) )
        Goto: Print_Error
    End_If

    Comment: Offer to let the user print additional labels.
    While( Ask_Yes_No( "Do you want to print the label again?", "Reprint",
TRUE ) )
        If_Not( Printer_Repeat )
            Goto: Print_Error
```

```
        End_If
    End_While

    Message( "Printing successful.", 5 )
    Return

Label: Print_Error
    Printer_Cancel
    Message( "Unable to print the label.", 5 )
    Return
```

## Printer\_Cancel

Instructs the printer to discard all Printer\_Data information already received.

---

### Example

```
Script( Printing_Example )
String( sScanData )
Number( sScanType )
Activate( On_Input, sScanData, sScanType )
    Comment: Prints a label containing displaying the barcode scanned.
    If_Not( Printer_Data( "! 0 200 200 148 1\OD\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "LABEL\OD\0A", FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "BARCODE CODE39 2 1 45 30 70 ", FALSE ) )
        Goto: Print_Error
    End_If
    If( Number_Greater_Than( String_Length( sScanData ), 20 ) )
        Printer_Cancel
        Message( "Barcode is too long.", 5 )
        Return
    End_If
    If_Not( Printer_Data( sScanData, FALSE ) )
        Goto: Print_Error
    End_If
    If_Not( Printer_Data( "\OD\0A", TRUE ) )
        Goto: Print_Error
    End_If

    Comment: Offer to let the user print additional labels.
    While( Ask_Yes_No( "Do you want to print the label again?", "Reprint",
TRUE ) )
        If_Not( Printer_Repeat )
            Goto: Print_Error
        End_If
    End_While

    Message( "Printing successful.", 5 )
    Return
```

```
Label: Print_Error  
  Printer_Cancel  
  Message( "Unable to print the label.", 5 )  
  Return
```

## String Values

This section contains a list of actions that return a string value. The following action categories are described in this section:

### Get System Information

| Action                                       | Description   |
|--|---|
| <a href="#">Get_MAC_Address</a>              | Gets the MAC address of the device.                               |
| <a href="#">Get_IP_Address</a>               | Gets the IP address of the device.                                |
| <a href="#">Get_Field_Symbology_ID</a>       | Gets the symbology ID of the specified field.                     |
| <a href="#">Get_Screen_Text</a>              | Gets the text at the specified location.                          |
| <a href="#">Get_Screen_Text_Length</a>       | Gets the specified amount of text on the screen.                  |
| <a href="#">Get_Screen_Text_Columns</a>      | Get text from a row on the screen, starting at a specific column. |
| <a href="#">Get_Workstation_ID</a>           | Gets the Workstation ID of the device.                            |
| <a href="#">Get_Avalanche_Property_Value</a> | Gets the value of the specified Avalanche property.               |

### Scanner Information

| Action                             | Description                     |
|------------------------------------|---------------------------------|
| <a href="#">Get_Scan_Type_Name</a> | Gets the name of the scan type. |

### Web Elements

| Action                                  | Description   |
|---|---|
| <a href="#">Web_Get_Source</a>          | Returns the HTML code of the search string.               |
| <a href="#">Web_Get_Current_Element</a> | Returns the HTML code for the Web element with the focus. |

### ESC Sequence Support

| Action                          | Description  |
|---------------------------------|--|
| <a href="#">Escape_Sequence</a> | Handles the supplied Wavelink Custom or Telxon ESC Sequence for all emulation types. |

## String Variable Assignments

| Action   | Description  |
|--|--|
| <a href="#">String_Set</a>                             | Gets the specified string.                                   |
| <a href="#">String_Combine</a>                         | Returns the concatenated value of two strings.               |
| <a href="#">String_Left</a>                            | Returns the specified characters of the input string.        |
| <a href="#">String_Right</a>                           | Returns the specified characters of the input string.        |
| <a href="#">String_Middle</a>                          | Returns the specified characters of the input string.        |
| <a href="#">String_Upper</a>                           | Converts the specified text to uppercase letters.            |
| <a href="#">String_Lower</a>                           | Converts the specified text to lowercase letters.            |
| <a href="#">String_Replace</a>                         | Replaces the specified text with another string.             |
| <a href="#">String_Only_Characters</a>                 | Gets a string with the specified characters.                 |
| <a href="#">String_Strip_Characters</a>                | Strips the specified characters from the string.             |
| <a href="#">String_Trim_Spaces_Start</a>               | Gets the specified text with all tabs and spaces deleted.    |
| <a href="#">String_Trim_Spaces_End</a>                 | Gets the specified text with all tabs and spaces deleted.    |
| <a href="#">Number_To_String_Binary</a>                | Gets the binary representation of the specified number.      |
| <a href="#">Number_To_String_Octal</a>                 | Gets the octal representation of the specified number.       |
| <a href="#">Number_To_String_Decimal</a>               | Gets the decimal representation of the specified number.     |
| <a href="#">Number_To_String_Hexadecimal_Lowercase</a> | Gets the hexadecimal representation of the specified number. |
| <a href="#">Number_To_String_Hexadecimal_Uppercase</a> | Gets the hexadecimal representation of the specified number. |
| <a href="#">Ask_String</a>                             | Displays a dialog box asking the user for a string.          |
| <a href="#">Ask_String_Password</a>                    | Displays a dialog box asking the user for a string.          |

---

| Action                               | Description   |
|--------------------------------------|---|
| <a href="#">Ask_String_Uppercase</a> | Displays a dialog box asking the user for a string. |
| <a href="#">Ask_String_Lowercase</a> | Displays a dialog box asking the user for a string. |

### **Number to Character Conversion**

| Action                              | Description   |
|-------------------------------------|---|
| <a href="#">Number_To_Character</a> | Converts the specified number to the character value. |

### **Field Identifiers and Data**

| Action                                     | Description                                     |
|--|---|
| <a href="#">Get_Field_Data_ID</a>          | Gets the Data ID for the specified field.       |
| <a href="#">Get_Field_Prefix_Scan_Data</a> | Gets the prefixed data for the specified field. |

## Get\_MAC\_Address

Get the MAC address of the device.

---

### Return Value

Returns the current MAC address for the device.

---

### Example

```
Script(Get_MAC_Address_Test)
String(strMacAddress)
Activate(From_Menu)
    strMacAddress = Get_MAC_Address
    Ask_OK(strMacAddress, "MacAddress")
Return
```

---

### See Also

[Get\\_IP\\_Address](#), [Get\\_Workstation\\_ID](#), [Get\\_Session\\_Number](#)

## Get\_IP\_Address

Get the IP address of the device.

---

### Return Value

Returns the current IP address for the device.

---

### Example

```
Script(Get_IP_Address_Test)
String(stripAddress)
Activate(From_Menu)
    stripAddress = Get_IP_Address
    Ask_OK(stripAddress, "IP Address")
Return
```

---

### See Also

[Get\\_MAC\\_Address](#), [Get\\_Workstation\\_ID](#), [Get\\_Session\\_Number](#)

## Get\_Field\_Symbology\_ID

Gets the symbology ID of the specified field.

---

### Parameters

|                        |                             |
|------------------------|-----------------------------|
| <i>Field Index</i>     | The index of the field.     |
| <i>Symbology Index</i> | The index of the symbology. |

---

### Format

Get\_Field\_Symbology\_ID (Field Index, Symbology Index)

---

### Return Value

Returns the field symbology ID.

---

### Remarks

There may be more than one symbology ID in a field; pass in the zero-based Symbology Index. For example, Symbology Index 0 gets the first symbology ID. The return ID ANY means Use All Symbologies. An empty or blank return ID means either the field has no symbology IDs or the field index is not valid. Use `Get_Num_Field_Symbology_Ids` to determine the number of symbologies for a field. This action is only valid when using IBM 5250 or 5555 emulation.

Scanner symbology values can be found in *Symbologies and Values* on page 322.

---

### Example

```
Script(Get_Field_Symbology_ID_Test)
String(strSymbologyID)
Boolean(ok)
Number(numSymbologies)
```

---

```
Number(counter)
Activate(From_Menu)
  Comment: Set some symbologies for field 0, then display them.
  ok = Set_Field_Symbology_ID(0, "UPCE0", FALSE)
  ok = Set_Field_Symbology_ID(0, "CODE 39", FALSE)
  ok = Set_Field_Symbology_ID(0, "EAN8", FALSE)
  numSymbologies = Get_Num_Field_Symbology_IDs(0)
  counter = 0
  While(Number_Less_Than(counter, numSymbologies))
    strSymbologyID = Get_Field_Symbology_ID(0, counter)
    Ask_OK(strSymbologyID, "Symbology for Field 0")
    counter = Number_Plus(counter, 1)
  End_While
Return
```

---

## See Also

[Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Get\\_Field\\_Data\\_ID](#)[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Num\\_Fields](#)

## Get\_Screen\_Text

Get the text at the specified location.

---

### Parameters

|               |                               |
|---------------|-------------------------------|
| <i>Row</i>    | The row of text to return.    |
| <i>Column</i> | The column of text to return. |

---

### Format

Get\_Screen\_Text (Row, Column)

---

### Return Value

Returns the text starting at the specified screen position up to the right side of the display.

---

### Remarks

The top row is row 1; the left-most column is column 1.

---

### Example

```
Script(Get_Screen_Text_Test)
String(strScreenText)
Number(nRow)
Number(nColumn)
Activate(From_Menu)
    nRow = Ask_Number("Enter row, top row is 1",
    "Get_Screen_Text_Test", 1, 99, 1)
    nColumn = Ask_Number("Enter column, left-most column is
    1", "Get_Screen_Text_Test", 1, 99, 1)
    strScreenText = Get_Screen_Text(nRow, nColumn)
```

```
Ask_OK(strScreenText, "Screen Text")  
Return
```

---

**See Also**

[Get\\_Screen\\_Text\\_Columns](#), [Get\\_Position\\_Row](#), [Get\\_Screen\\_Text\\_Length](#),  
[Get\\_Field\\_Index](#), [Get\\_Field\\_Row](#), [String\\_Equal](#), [Ask\\_String](#), [Get\\_Screen\\_Rows](#),  
[Search\\_Screen](#), [Speech\\_To\\_Text](#)

## Get\_Screen\_Text\_Length

Get the specified amount of text on the screen.

---

### Parameters

|                       |  |
|-----------------------|--|
| <i>Row</i>            | The row of text to return.                               |
| <i>Column</i>         | The column of text to return.                            |
| <i>Maximum Length</i> | The maximum number of characters to get from the screen. |

---

### Format

```
Get_Screen_Text_Length (Row, Column, Maximum Length)
```

---

### Return Value

Returns the text starting at the specified screen position up to the right side of the display.

---

### Remarks

The string will be truncated if it is longer than the number of characters specified. The top row is row 1; the left-most column is 1.

---

### Example

```
Script(Get_Screen_Text_Length_Test)
String(strScreenText)
Number(nRow)
Number(nColumn)
Number(nMaxCharacters)
Activate(From_Menu)
    nRow = Ask_Number("Enter row, top row is 1",
```

```
"Get_Screen_Text_Test", 1, 99, 1)
nColumn = Ask_Number("Enter column, top column is 1",
"Get_Screen_Text_Test", 1, 99, 1)
nMaxCharacters = Ask_Number("Enter maximum text length",
"Get_Screen_Text_Test", 1, 99, 25)
strScreenText = Get_Screen_Text_Length(nRow, nColumn,
nMaxCharacters)
Ask_OK(strScreenText, "ScreenText")
Return
```

---

## See Also

[Speech\\_To\\_Text](#), [Search\\_Screen](#), [Get\\_Field\\_Index](#), [Get\\_Field\\_Row](#), [Get\\_Screen\\_Text](#),  
[Get\\_Screen\\_Text\\_Columns](#), [Get\\_Screen\\_Rows](#), [Get\\_Position\\_Row](#), [String\\_Equal](#),  
[Ask\\_String](#)

## Get\_Screen\_Text\_Columns

Get text from a row on the screen, limited by the specified number of columns.

---

### Parameters

|                          |   |
|--------------------------|---|
| <i>Row</i>               | The row of text to return.                            |
| <i>Column</i>            | The column of text to return.                         |
| <i>Number of Columns</i> | The maximum number of columns to get from the screen. |

---

### Format

```
Get_Screen_Text_Columns (Row, Column, Number of Columns)
```

---

### Return Value

Returns the text starting at the specified screen position up to the right side of the display.

---

### Remarks

The string will not include information past the number of columns specified.

---

### Example

```
Script( Get_Screen_Text_Columns_Test )
String( strScreenText )
Number( nRow )
Number( nColumn )
Number( nMaxColumns )
Activate( From_Menu )
    nRow = Ask_Number( "Enter row, top row is 1", "Get_Screen_Text_Test",
1,
```

```
99, 1 )
    nColumn = Ask_Number( "Enter column, top column is 1",
"Get_Screen_Text_Test", 1, 99, 1 )
    nMaxColumns = Ask_Number( "Enter maximum number of columns",
"Get_Screen_Text_Test", 1, 99, 10 )
    strScreenText = Get_Screen_Text_Columns( nRow, nColumn, nMaxColumns )
    Ask_OK( strScreenText, "Screen Text" )
Return
```

---

## See Also

[Get\\_Screen\\_Text](#), [Get\\_Screen\\_Text\\_Length](#), [Get\\_Field\\_Index](#), [Get\\_Field\\_Row](#),  
[Speech\\_To\\_Text](#), [Search\\_Screen](#), [Get\\_Screen\\_Rows](#), [Get\\_Position\\_Row](#), [Ask\\_String](#),  
[String\\_Equal](#)

## Get\_Workstation\_ID

Get the Workstation ID of the device.

---

### Return Value

Returns the current Workstation ID.

---

### Remarks

This is only valid when using IBM emulation (3270, 5250, or 5555) and a Workstation ID has been specified for the current Host Profile. Otherwise, an empty string is returned.

---

### Example

```
Script(Get_Workstation_ID_Test)
String(strWorkstationID)
    strWorkstationID = Get_Workstation_ID
    Ask_OK(strWorkstationID, "Workstation ID")
Return
```

---

### See Also

[Get\\_MAC\\_Address](#), [Get\\_IP\\_Address](#), [Get\\_Session\\_Number](#)



## Get\_Scan\_Type\_Name

Get the name of the scan type.

---

### Parameters

*Scan Type*                      The scan type number.

---

### Format

Get\_Scan\_Type\_Name (Scan Type)

---

### Return Value

Returns the name of the supplied scan type.

---

### Remarks

An empty string is returned if the scan type is not recognized. Scanner symbology values can be found in *Symbologies and Values* on page 322.

---

### Example

```
Script( Get_Scan_Type_Name_Test )
String( barcode )
String( strScanType )
Number( type )
Activate( On_Input, barcode, type )
    strScanType = Get_Scan_Type_Name( type )
    Ask_OK( strScanType, "Scan Type Name" )
    type = Ask_Number( "Enter a scan type", "Get_Screen_Text_Test", 0,
255,
60 )
    strScanType = Get_Scan_Type_Name( type )
```

```
Ask_OK( strScanType, "Scan Type Name" )  
Return
```

---

**See Also**

[Scan\\_String](#), [Get\\_Scan\\_Type\\_Value](#)

## Web\_Get\_Source

Returns the HTML code immediately following the first instance of the search string or an empty string if not found.

---

### Parameters

|               |   |
|---------------|---|
| <i>Value1</i> | The Search String.  |
| <i>Value2</i> | The maximum number of characters to return (integer).         |
| <i>Value3</i> | TRUE to ignore the letter case. FALSE if otherwise. Boolean). |
| <i>Value4</i> | TRUE to search the frames of the page. FALSE if otherwise..   |

---

### Format

```
Web_Get_Source ("Value1", Value2, Value3, Value4)
```

---

### Remarks

Returns the start of the page if Search String is blank.

If the maximum length is 0, all the data that will fit in the string is returned.

---

### Example

```
Script( Web_Get_Source_First )
String( strSource )
Activate( From_Menu )
    strSource = Web_Get_Source( "", 0, FALSE, FALSE )
    Ask_OK( strSource, "Web Page Text" )
Return
```

## Web\_Get\_Current\_Element

Returns the HTML code for the Web element with the focus. Returns an empty string if no Web element has the focus.

---

### Format

Web\_Get\_Source

---

### Example

```
Script( Web_Get_Current_Element_Test )
String( strCurrentElement )
Activate( From_Menu )
    strCurrentElement = Web_Get_Current_Element
    Ask_OK( strCurrentElement, "Web Page Current Element Text" )
Return
```



---

# String\_Set

Assign a string value.

---

## Parameters

*String*                      The String, Variable or Action that returns a string.

---

## Format

String\_Set (String)

---

## Return Value

Returns the value of the string.

---

## Remarks

The equal sign (=) is the symbol for `String_Set` in the Script Editor.

---

## Example

```
Script(String_Set_Test)
String(strResult)
String(strTitle)
Activate(From_Menu)
    strTitle = "Scripting String"
    strResult = Ask_String("Enter some text",
        "String_Set_Test", 1, 99, "")
    strResult = String_Combine("Text Entered:", strResult)
```

```
Ask_OK(strResult, strTitle)
Return
```

---

**See Also**

[String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#),  
[String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

---

# String\_Combine

Returns the concatenated value of two strings.

---

## Parameters

*String1*                      The first part of the returned string.

*String2*                      The second part of the returned string.

---

## Format

String\_Combine (String1, String2)

---

## Return Value

Returns the value of string1 concatenated with string2.

---

## Example

```
Script(String_Set_Test)
String(strResult)
String(strTitle)
Activate(From_Menu)
    strTitle = "Scripting String"
    strResult = Ask_String("Enter some text",
        "String_Set_Test", 1, 99, "")
    strResult = String_Combine("Text Entered:", strResult)
    Ask_OK(strResult, strTitle)
Return
```

---

## See Also

[String\\_Set](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#),  
[String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),

String\_Trim\_Spaces\_Start, String\_Trim\_Spaces\_End, Number\_To\_String\_Binary,  
Number\_To\_String\_Octal, Number\_To\_String\_Decimal,  
Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal

---

## String\_Left

Returns the specified characters of the input string.

---

### Parameters

|                             |   |
|-----------------------------|---|
| <i>String</i>               | The string from which to get the characters.                              |
| <i>Number of Characters</i> | The number of characters to get, starting at the beginning of the string. |

---

### Format

String\_Left (String, Number of Characters)

---

### Return Value

Returns a string with just the first n characters of the input string. If the input string is less than n characters, the entire string is returned.

---

### Example

```
Script(String_Left_Test)
String(strEntered)
String(strLeftPart)
Activate(From_Menu)
    strEntered = Ask_String("Enter some text",
        "String_Left_Test", 1, 99, "")
    strLeftPart = String_Left(strEntered, 2)
    Ask_OK(strLeftPart, "String_Left 2 characters")
Return
```

---

### See Also

String\_Set, String\_Combine, String\_Right, String\_Middle, String\_Upper, String\_Lower,  
String\_Replace, String\_Only\_Characters, String\_Strip\_Characters,  
String\_Trim\_Spaces\_Start, String\_Trim\_Spaces\_End, Number\_To\_String\_Binary,  
Number\_To\_String\_Octal, Number\_To\_String\_Decimal,  
Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal

---

# String\_Right

Returns the specified characters of the input string.

---

## Parameters

|                 |   |
|-----------------|---|
| <i>String</i>   | The string from which to get the characters.                              |
| <i>Sequence</i> | The number of characters to get, starting at the beginning of the string. |

---

## Format

String\_Right (String, Sequence)

---

## Return Value

Returns a string with just the last *n* characters of the input string. If the input string is less than *n* characters, the entire string is returned.

---

## Example

```
Script(String_Right_Test)
String(strEntered)
String(strRightPart)
Activate(From_Menu)
    strEntered = Ask_String("Enter some text", "String_Right_Test", 1, 99,
    "")
    strRightPart = String_Right(strEntered, 3)
    Ask_OK(strRightPart, "String_Right 3 characters")
Return
```

---

## See Also

String\_Set, String\_Combine, String\_Left, String\_Middle, String\_Upper, String\_Lower,  
String\_Replace, String\_Only\_Characters, String\_Strip\_Characters,  
String\_Trim\_Spaces\_Start, String\_Trim\_Spaces\_End, Number\_To\_String\_Binary,  
Number\_To\_String\_Octal, Number\_To\_String\_Decimal,  
Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal

---

## String\_Middle

Returns the specified characters of the input string.

---

### Parameters

|                             |   |
|-----------------------------|---|
| <i>String</i>               | The string from which to get the characters.                |
| <i>Starting Position</i>    | The character with which to start.                          |
| <i>Number of Characters</i> | The number of characters to get from the end of the string. |

---

### Format

`String_Middle (String, Starting Position, Number of Characters)`

---

### Return Value

Returns a string with just the middle *n* characters of the input string. If the input string is less than *n* characters, the entire string is returned.

---

### Remarks

The string parsing starts at the position specified, with 0 being the left-most character, so a position value of 0 is the same as `String_Left`.

---

### Example

```
Script(String_Middle_Test)
String(strEntered)
String(strMiddlePart)
Number(nStart)
Number(nCharacters)
Activate(From_Menu)
```

```
strEntered = Ask_String("Enter some text",  
"String_Middle_Test", 1, 99, "")  
nStart = Ask_Number("Enter start characters, zero is  
the first character", "String_Middle_Test", 0, 99, 2)  
nCharacters = Ask_Number("Enter number of characters",  
"String_Middle_Test", 0, 99, 3)  
strMiddlePart = String_Middle(strEntered, nStart,  
nCharacters)  
Ask_OK(strMiddlePart, "String_Middle")  
Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Upper](#), [String\\_Lower](#),  
[String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

## String\_Upper

Converts the specified text to uppercase letters.

---

### Parameters

*String*                                      The string to convert to uppercase.

---

### Format

String\_Upper (String)

---

### Return Value

Returns a string with all characters converted to uppercase.

---

### Example

```
Script(String_Upper_Lower_Test)
String(strEntered)
String(strUpper)
String(strLower)
Activate(From_Menu)
    strEntered = Ask_String("Enter some text",
        "String_Upper_Lower_Test", 1, 99, "")
    strUpper = String_Upper(strEntered)
    Ask_OK(strUpper, "String_Upper_Test")
    strLower = String_Lower(strEntered)
    Ask_OK(strLower, "String_Lower_Test")
Return
```

---

### See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),

String\_Trim\_Spaces\_Start, String\_Trim\_Spaces\_End, Number\_To\_String\_Binary,  
Number\_To\_String\_Octal, Number\_To\_String\_Decimal,  
Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal

# String\_Lower

Converts the specified text to lowercase letters.

---

## Parameters

*String*                                      The string to convert to lowercase.

---

## Format

String\_Lower (String)

---

## Return Value

Returns a string with all characters converted to lowercase.

---

## Example

```
Script(String_Upper_Lower_Test)
String(strEntered)
String(strUpper)
String(strLower)
Activate(From_Menu)
    strEntered = Ask_String("Enter some text",
        "String_Upper_Lower_Test", 1, 99, "")
    strUpper = String_Upper(strEntered)
    Ask_OK(strUpper, "String_Upper_Test")
    strLower = String_Lower(strEntered)
    Ask_OK(strLower, "String_Lower_Test")
Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),

String\_Trim\_Spaces\_Start, String\_Trim\_Spaces\_End, Number\_To\_String\_Binary,  
Number\_To\_String\_Octal, Number\_To\_String\_Decimal,  
Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal

# String\_Replace

Replaces the specified text with another string.

---

## Parameters

|                              |  |
|------------------------------|--|
| <i>String to Parse</i>       | The original string that gets the substring replaced.                  |
| <i>Substring to Replace</i>  | Find all instances of this and replace them.                           |
| <i>Replacement Substring</i> | The replacement text to use.   |
| <i>Ignore Case</i>           | Indicates whether the case of the letters is taken into consideration. |

---

## Format

```
String_Replace ("String to Parse", "Substring to Replace", "Replacement Substring", Ignore Case)
```

---

## Return Value

Returns a string where all instances of *Substring to Replace* have been replaced with *Replacement Substring*.

---

## Remarks

If *Ignore Case* is TRUE then upper-case and lower-case letters are considered to be equal.

---

## Example

```
Script( String_Replace_Test )
String( strResult )
Activate( From_Menu )
    strResult = String_Replace( "123456789012345", "2", "aaaaa", FALSE )
    Message( strResult, 0 )
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

---

## String\_Only\_Characters

Gets a string with the specified characters.

---

### Parameters

|                           |  |
|---------------------------|--|
| <i>String to Parse</i>    | The original string that gets stripped of all characters except those specified. |
| <i>Characters to Keep</i> | The characters that are not stripped from the original string.                   |
| <i>Ignore Case</i>        | Indicates whether the case of the letters is taken into consideration.           |

---

### Format

`String_Only_Characters (String to Parse, Characters to Keep, Ignore Case)`

---

### Return Value

Returns a string where all characters in `String to Parse` that are not in `Characters to Keep` have been deleted.

---

### Remarks

If `Ignore Case` is TRUE then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script(String_Only_Characters_Test)
String(strEntered)
String(strResult)
String(strCharactersToKeep)
Activate(From_Menu)
```

```
strEntered = Ask_String("Enter some text",  
"String_Only_Characters_Test", 1, 99, "abcdefghijkl")  
strCharactersToKeep = Ask_String("Enter the characters to keep",  
"String_Only_Characters_Test", 1, 99, "abc")  
strResult = String_Only_Characters(strEntered,  
strCharactersToKeep, TRUE)  
Ask_OK(strResult, "String_Only_Characters_Test")  
Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#),  
[String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#),  
[Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

---

## String\_Strip\_Characters

Strips the specified characters from the string.

---

### Parameters

|                            |  |
|----------------------------|--|
| <i>String to Parse</i>     | The original string that gets stripped of all characters except those specified. |
| <i>Characters to Strip</i> | The characters to be stripped from the original string.                          |
| <i>Ignore Case</i>         | Indicates whether the case of the letters is taken into consideration.           |

---

### Format

`String_Strip_Characters (String to Parse, Characters to Strip, Ignore Case)`

---

### Return Value

Returns a string where all characters in `String to Parse` that are not in `Characters to Keep` have been deleted.

---

### Remarks

If `Ignore Case` is TRUE then upper-case and lower-case letters are considered to be equal.

---

### Example

```
Script( String_Strip_Characters_Test )
String( strEntered )
String( strResult )
String( strCharactersToStrip )
```

```
Activate( From_Menu )
    strEntered = Ask_String( "Enter some text",
"String_Strip_Characters_Test", 1, 99, "abcdefghijkl" )
    strCharactersToStrip = Ask_String( "Enter the characters to strip",
"String_Strip_Characters_Test", 1, 99, "abc" )
    strResult = String_Strip_Characters( strEntered,
strCharactersToStrip,
TRUE )
    Ask_OK( strResult, "String_Strip_Characters_Test" )
Return
```

---

## **See Also**

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#),  
[String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#),  
[Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

---

## String\_Trim\_Spaces\_Start

Gets the specified text with all tabs and spaces deleted from the beginning.

---

### Parameters

|                        |   |
|------------------------|---|
| <i>String to Parse</i> | The original string which is removed of all spaces and tabs at the beginning. |
|------------------------|---|

---

### Format

```
String_Trim_Spaces_Start ("String to Parse")
```

---

### Return Value

Returns a string where all spaces and tabs at the start of the string have been deleted.

---

### Example

```
Script( String_Trim_Spaces_Start_Test )
String( strResult )
Activate( From_Menu )
    strResult = String_Trim_Spaces_Start( "      567890" )
    Message( strResult, 0 )
Return
```

---

### See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#), [Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

## String\_Trim\_Spaces\_End

Gets the specified text with all tabs and spaces deleted from the end.

---

### Parameters

*String to Parse*                      The original string which is removed of all spaces and tabs at the end.

---

### Format

String\_Trim\_Spaces\_End (String to Parse)

---

### Return Value

Returns a string where all spaces and tabs at the end of the string have been deleted.

---

### Example

```
Script( String_Trim_Spaces_End_Test )
String( strResult )
String( strOriginal )
String( strMessage )
Activate( From_Menu )
    strOriginal = "abcd   "
    strResult = String_Trim_Spaces_End( strOriginal )
    strMessage = String_Combine( "", strOriginal )
    strMessage = String_Combine( strMessage, "" converted to "" )
    strMessage = String_Combine( strMessage, strResult )
    strMessage = String_Combine( strMessage, "" )
    Message( strMessage, 8 )
Return
```

---

### See Also

---

String\_Set, String\_Combine, String\_Left, String\_Right, String\_Middle, String\_Upper,  
String\_Lower, String\_Replace, String\_Only\_Characters, String\_Strip\_Characters,  
String\_Trim\_Spaces\_Start, Number\_To\_String\_Binary, Number\_To\_String\_Octal,  
Number\_To\_String\_Decimal, Number\_To\_String\_Hexadecimal\_Lowercase,  
Number\_To\_String\_Hexadecimal\_Uppercase, Ask\_String, Ask\_String\_Password,  
Ask\_String\_Uppercase, Ask\_String\_Lowercase, String\_Equal



```
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of",
strEntered))
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#), [Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)



```
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of ",
strEntered))
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#), [Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)



```
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of ",
strEntered))
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#), [Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)



```
    strEntered)
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of ",
    strEntered))
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)



```
    strEntered)
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of ",
    strEntered))
    Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#), [String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#), [String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#), [Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#), [Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

# Ask\_String

Displays a dialog box asking the user for a string.

---

## Parameters

|                                 |  |
|---------------------------------|--|
| <i>Message Text</i>             | The message displayed in the dialog box.                               |
| <i>Title Text</i>               | The title displayed in the dialog box.                                 |
| <i>Minimum Length of String</i> | The minimum number of characters the string must have.                 |
| <i>Maximum Length of String</i> | The maximum number of characters the string can have.                  |
| <i>Default String</i>           | The initial value in the message box which can be changed by the user. |

---

## Format

`Ask_String ("Enter a string", "Ask_String_Test", Minimum Length of String, Maximum Length of String, Default String)`

---

## Return Value

Returns the string supplied by the user.

---

## Remarks

The supplied default string is returned (unaltered) if the user cancels the dialog.

---

## Example

```
Script(Ask_String_Test)
String(strEntered)
Activate(From_Menu)
    strEntered = Ask_String("Enter a string",
        "Ask_String_Test", 1, 99, "")
    Ask_OK(strEntered, "You Entered")
Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

# Ask\_String\_Password

Displays a dialog box asking the user for a string.

---

## Parameters

|                                 |  |
|---------------------------------|--|
| <i>Message Text</i>             | The message displayed in the dialog box.                               |
| <i>Title Text</i>               | The title displayed in the dialog box.                                 |
| <i>Minimum Length of String</i> | The minimum number of characters the string must have.                 |
| <i>Maximum Length of String</i> | The maximum number of characters the string can have.                  |
| <i>Default String</i>           | The initial value in the message box which can be changed by the user. |

---

## Format

```
Ask_String_Password ("Enter a string", "Ask_String_Test", Minimum Length  
of String, Maximum Length of String, Default String)
```

---

## Return Value

Returns the string supplied by the user.

---

## Remarks

The string is displayed as a password (a series of asterisks).

---

The supplied default string is returned (unaltered) if the user cancels the dialog.

---

## Example

```
Script(Ask_String_Password_Test)
String(strEntered)
Activate(From_Menu)
    strEntered = Ask_String_Password("Enter a password",
    "Ask_String_Password_Test", 1, 99, "")
    Ask_OK(strEntered, "The password is")
Return
```

---

## See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Uppercase](#),  
[Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

## Ask\_String\_Uppercase

Displays a dialog box asking the user for a string.

---

### Parameters

|                                 |  |
|---------------------------------|--|
| <i>Message Text</i>             | The message displayed in the dialog box.                               |
| <i>Title Text</i>               | The title displayed in the dialog box.                                 |
| <i>Minimum Length of String</i> | The minimum number of characters the string must have.                 |
| <i>Maximum Length of String</i> | The maximum number of characters the string can have.                  |
| <i>Default String</i>           | The initial value in the message box which can be changed by the user. |

---

### Format

```
Ask_String_Uppercase ("Enter a string", "Ask_String_Test", Minimum Length  
of String, Maximum Length of String, Default String)
```

---

### Return Value

Returns the string supplied by the user.

---

### Remarks

Any lowercase letters are converted to uppercase characters.

---

The supplied default string is returned (unaltered) if the user cancels the dialog.

---

### Example

```
Script(Ask_String_Uppercase_Test)
String(strEntered)
Activate(From_Menu)
    strEntered = Ask_String_Uppercase("The string you enter
    will be upper case", "Ask_String_Uppercase", 1, 99, "")
    Ask_OK(strEntered, "The upper case string")
Return
```

---

### See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Lowercase](#), [String\\_Equal](#)

## Ask\_String\_Lowercase

Displays a dialog box asking the user for a string.

---

### Parameters

|                                 |  |
|---------------------------------|--|
| <i>Message Text</i>             | The message displayed in the dialog box.                               |
| <i>Title Text</i>               | The title displayed in the dialog box.                                 |
| <i>Minimum Length of String</i> | The minimum number of characters the string must have.                 |
| <i>Maximum Length of String</i> | The maximum number of characters the string can have.                  |
| <i>Default String</i>           | The initial value in the message box which can be changed by the user. |

---

### Format

`Ask_String_Lowercase ("Enter a string", "Ask_String_Test", Minimum Length of String, Maximum Length of String, Default String)`

---

### Return Value

Returns the string supplied by the user.

---

### Remarks

Any uppercase letters are converted to lowercase characters.

---

The supplied default string is returned (unaltered) if the user cancels the dialog.

---

### Example

```
Script(Ask_String_Lowercase_Test)
String(strEntered)
Activate(From_Menu)
    strEntered = Ask_String_Lowercase("The string you enter
    will be lower case", "Ask_String_Lowercase", 1, 99, "")
    Ask_OK(strEntered, "The lower case string")
Return
```

---

### See Also

[String\\_Set](#), [String\\_Combine](#), [String\\_Left](#), [String\\_Right](#), [String\\_Middle](#), [String\\_Upper](#),  
[String\\_Lower](#), [String\\_Replace](#), [String\\_Only\\_Characters](#), [String\\_Strip\\_Characters](#),  
[String\\_Trim\\_Spaces\\_Start](#), [String\\_Trim\\_Spaces\\_End](#), [Number\\_To\\_String\\_Binary](#),  
[Number\\_To\\_String\\_Octal](#), [Number\\_To\\_String\\_Decimal](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Lowercase](#),  
[Number\\_To\\_String\\_Hexadecimal\\_Uppercase](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [String\\_Equal](#)



```
Ask_OK( strCharacter, strTitle )  
Return
```

## Get\_Field\_Data\_ID

Gets the Data ID for the specified field.

---

### Parameters

|                      |                           |
|----------------------|---------------------------|
| <i>Field Index</i>   | The index of the field.   |
| <i>Data ID Index</i> | The index of the Data ID. |

---

### Format

`Get_Field_Data_ID (Field Index, Data ID Index)`

---

### Return Value

Returns the field's Data ID.

---

### Remarks

A blank string means no Data ID is set for the field or the field index is invalid. A field may have more than one Data ID. Use `Get_Num_Field_Data_IDs` to determine the number of Data IDs for a field.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

---

### Example

```
Script( Get_Field_Data_ID_Test )
String( strDataID )
Boolean( bSetOK )
Number( numDataIDs )
Number( counter )
```

```
Activate( From_Menu )
    bSetOK = Set_Field_Data_ID( 0, "N" )
    numDataIDs = Get_Num_Field_Data_IDs( 0 )
    counter = 0
    While( Number_Less_Than( counter, numDataIDs ) )
        strDataID = Get_Field_Data_ID( 0, counter )
        Ask_OK( strDataID, "Data ID for Field 0" )
        counter = Number_Plus( counter, 1 )
    End_While
Return
```

---

## See Also

[Get\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#),  
[Get\\_Field\\_Symbology\\_ID](#), [Get\\_Num\\_Fields](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Set\\_Field\\_Data\\_ID](#), [Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#)

## Get\_Field\_Prefix\_Scan\_Data

Gets the prefixed data for the specified field.

---

### Parameters

*Field Index*                      The index of the field.

---

### Format

Get\_Field\_Prefix\_Scan\_Data (Field Index)

---

### Return Value

Returns the data prefixed when the field is scanned.

---

### Remarks

This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Prefix_Scan_Data_Test)
String(strPrefix)
Boolean(bOK)
Activate(From_Menu)
    bOK = Set_Field_Prefix_Scan_Data(0, "SCAN")
    strPrefix = Get_Field_Prefix_Scan_Data(0)
    Ask_OK(strPrefix, "String that will be prefixed to scan
    data in field")
Return
```

---

### See Also

Get\_Field\_Data\_ID, Get\_Field\_Prefix\_Scan\_Data, Get\_Num\_Field\_Data\_IDs,  
Get\_Num\_Field\_Symbology\_IDs, Get\_Field\_Com\_Data\_Field,  
Get\_Field\_Symbology\_ID, Get\_Num\_Fields, Set\_Field\_Symbology\_ID,  
Set\_Field\_Data\_ID, Get\_Field\_Symbology\_Operator, Set\_Field\_Append\_Scan\_Data,  
Set\_Field\_Com\_Data\_Field, Set\_Field\_Prefix\_Scan\_Data, Get\_Field\_Append\_Scan\_Data

## Integer Values

This section contains a list of actions that return an integer value. The following action categories are described in this section:

### Get System Information

| Action                                      | Description  |
|---|--|
| <a href="#">Get_Screen_Columns</a>          | Gets the number of columns on the screen.                        |
| <a href="#">Get_Screen_Rows</a>             | Gets the number of rows on the screen.                           |
| <a href="#">Get_Position_Column</a>         | Gets the column number on which the cursor is currently located. |
| <a href="#">Get_Position_Row</a>            | Get the row number on which the cursor is currently located.     |
| <a href="#">Get_Session_Number</a>          | Get the number for the session in which this script is executing |
| <a href="#">Get_Time</a>                    | Get the amount of time passed since January 1, 2000.             |
| <a href="#">Get_Time_Since_Reset</a>        | Gets the amount of time since the last reboot.                   |
| <a href="#">Get_Field_Index</a>             | Get the index of a field at the specified row and column.        |
| <a href="#">Get_Num_Fields</a>              | Get the number of fields on the screen.                          |
| <a href="#">Get_Field_Index_Row_Text</a>    | Get the index of a field that is in the same row as the text.    |
| <a href="#">Get_Field_Index_Column_Text</a> | Get the index of a field that is in the same column as the text. |
| <a href="#">Get_Field_Row</a>               | Get the row number of the field.                                 |
| <a href="#">Get_Field_Column</a>            | Get the column number of the field.                              |
| <a href="#">Get_Field_Length</a>            | Get the length of the field.                                     |
| <a href="#">Get_Num_Field_Data_IDs</a>      | Get the number of data IDs in a field.                           |
| <a href="#">Get_Num_Field_Symbology_IDs</a> | Get the number of symbology IDs in a field.                      |
| <a href="#">Get_Field_Com_Data_Field</a>    | Get the index of the field that is the Com Data Field.           |

### Field Identifiers and Data

| Action                          | Description   |
|---------------------------------|---|
| <a href="#">Get_Field_Index</a> | Get the index of a field at the specified row and column. |
| <a href="#">Get_Num_Fields</a>  | Get the number of fields on the screen.                   |

---

| Action                                      | Description  |
|---|--|
| <a href="#">Get_Field_Index_Row_Text</a>    | Get the index of a field that is in the same row as the text.    |
| <a href="#">Get_Field_Index_Column_Text</a> | Get the index of a field that is in the same column as the text. |
| <a href="#">Get_Field_Row</a>               | Get the row number of the field.                                 |
| <a href="#">Get_Field_Column</a>            | Get the column number of the field.                              |
| <a href="#">Get_Field_Length</a>            | Get the length of the field.                                     |
| <a href="#">Get_Num_Field_Data_IDs</a>      | Get the number of data IDs in a field.                           |
| <a href="#">Get_Num_Field_Symbology_IDs</a> | Get the number of symbology IDs in a field.                      |
| <a href="#">Get_Field_Com_Data_Field</a>    | Get the index of the field that is the Com Data Field.           |

## Scanner Information

| Action                              | Description  |
|-------------------------------------|--|
| <a href="#">Get_Scan_Type_Value</a> | Get the number value of the supplied scan type name. |

## General Queries

| Action                            | Description   |
|-----------------------------------|---|
| <a href="#">Ask_Yes_No_Cancel</a> | Displays a message in a dialog box with a Yes, No, and Cancel button and waits until the user selects a button. |
| <a href="#">Run_Application</a>   | Starts an application with the flags (optional)   |

## String Handling

| Action                            | Description  |
|-----------------------------------|--|
| <a href="#">String_Length</a>     | Get the number of characters in a string.  |
| <a href="#">String_Find_First</a> | Finds the first instance of the substring inside the string, and returns the position where that substring starts. |
| <a href="#">String_Find_Last</a>  | Finds the last instance of the substring inside the string, and returns the position where that substring starts.  |

## Integer Assignments

| Action                                  | Description   |
|---|---|
| <a href="#">Number_Set</a>              | Set the value of a number variable.   |
| <a href="#">Number_Plus</a>             | Add two numbers together and return the sum.  |
| <a href="#">Number_Minus</a>            | Subtract the second term from the first term to get the difference.   |
| <a href="#">Number_Multiply</a>         | Multiply the first term by the second term and returns the product.   |
| <a href="#">Number_Divide</a>           | Divide the first term by the second term and return the product.  |
| <a href="#">Number_Divide_Remainder</a> | Divide the first term by the second term and return the remainder. For example, 7 divided by 3 would return a remainder of 1. |

## Convert Strings to Integers

| Action                                       | Description                                 |
|--|---|
| <a href="#">String_To_Number_Binary</a>      | Get a string's binary representation.       |
| <a href="#">String_To_Number_Octal</a>       | Gets a string's octal representation.       |
| <a href="#">String_To_Number_Decimal</a>     | Gets a string's decimal representation.     |
| <a href="#">String_To_Number_Hexadecimal</a> | Gets a string's hexadecimal representation. |

## Ask User for Integer

| Action                     | Description   |
|----------------------------|---|
| <a href="#">Ask_Number</a> | Displays a dialog box asking the user for a decimal number. |

## Number/Character Conversion

| Action                              | Description  |
|-------------------------------------|--|
| <a href="#">Character_To_Number</a> | Converts the character at position Index in the string into the number value for that character. |

## Bitwise Arguments

| Action                      | Description  |
|-----------------------------|--|
| <a href="#">Bitwise_And</a> | The resulting number will have a bit set when both input numbers have that bit set.                          |
| <a href="#">Bitwise_Or</a>  | The resulting number will have a bit set when either input numbers has that bit set (inclusive or).          |
| <a href="#">Bitwise_Xor</a> | The resulting number will have a bit set when exactly one input number has that bit set (exclusive or).      |
| <a href="#">Bitwise_Not</a> | The resulting number will have a bit set when the input number does not have that bit set (ones complement). |

## Get\_Screen\_Columns

Gets the number of columns on the screen. This is the total number of columns, not the number of columns visible.

---

### Example

```
Script (Screen_Info)
String (StrMessage)
Number (nColumns)
Number (nRows)
Number (nPositionRow)
Number (nPositionColumn)
Activate (From_Menu)
    nRows = Get_Screen_Rows
    nColumns = Get_Screen_Columns
    nPositionRow = Get_Position_Row
    nPositionColumn = Get_Position_Column
    strMessage = String_Combine ("Screen:",
    Number_To_String_Decimal (nRows)
    strMessage = String_Combine (strMessage, "rows,")
    strMessage = String_Combine (strMessage,
    Number_To_String_Decimal (nColumns)
    strMessage = String_Combine (strMessage, "columns")
    Message (strMessage, 10)
    strMessage = String_Combine ("Cursor position: row",
    Number_To_String_Decimal (nPositionRow)
    strMessage = String_Combine (strMessage, " column")
    strMessage = String_Combine (strMessage,
    Number_To_String_Decimal (nPositionColumn)
    Ask_OK (strMessage, "Screen_Info")
Return
```

---

### See Also

[Get\\_Screen\\_Rows](#), [Get\\_Position\\_Column](#), [Get\\_Position\\_Row](#), [Get\\_Time](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Column](#), [Set\\_Cursor\\_Position](#)

---

## Get\_Screen\_Rows

Get the number of rows on the screen. This is the total number of rows, not the number of rows visible.

---

### Example

```
Script(Screen_Info)
String(StrMessage)
Number(nColumns)
Number(nRows)
Number(nPositionRow)
Number(nPositionColumn)
Activate(From_Menu)
    nRows = Get_Screen_Rows
    nColumns = Get_Screen_Columns
    nPositionRow = Get_Position_Row
    nPositionColumn = Get_Position_Column
    strMessage = String_Combine("Screen:",
    Number_To_String_Decimal(nRows))
    strMessage = String_Combine(strMessage, "rows,")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nColumns))
    strMessage = String_Combine(strMessage, "columns")
    Message(strMessage, 10)
    strMessage = String_Combine("Cursor position: row",
    Number_To_String_Decimal(nPositionRow))
    strMessage = String_Combine(strMessage, " column")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nPositionColumn))
    Ask_OK(strMessage, "Screen_Info")
Return
```

---

### See Also

[Get\\_Screen\\_Columns](#), [Get\\_Position\\_Column](#), [Get\\_Position\\_Row](#), [Get\\_Field\\_Row](#),  
[Get\\_Field\\_Column](#), [Set\\_Cursor\\_Position](#)

## Get\_Position\_Column

Get the column number on which the cursor is currently located.

---

### Remarks

The left-most column is 1.

---

### Example

```
Script(Screen_Info)
String(StrMessage)
Number(nColumns)
Number(nRows)
Number(nPositionRow)
Number(nPositionColumn)
Activate(From_Menu)
    nRows = Get_Screen_Rows
    nColumns = Get_Screen_Columns
    nPositionRow = Get_Position_Row
    nPositionColumn = Get_Position_Column
    strMessage = String_Combine("Screen:",
    Number_To_String_Decimal(nRows))
    strMessage = String_Combine(strMessage, "rows,")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nColumns))
    strMessage = String_Combine(strMessage, "columns")
    Message(strMessage, 10)
    strMessage = String_Combine("Cursor position: row",
    Number_To_String_Decimal(nPositionRow))
    strMessage = String_Combine(strMessage, " column")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nPositionColumn))
    Ask_OK(strMessage, "Screen_Info")
Return
```

---

### See Also

Get\_Screen\_Columns, Get\_Screen\_Rows, Get\_Position\_Row, Set\_Cursor\_Position,  
Get\_Field\_Row, Get\_Field\_Column

## Get\_Position\_Row

Get the row number on which the cursor is currently located.

---

### Remarks

The top-most row is 1.

---

### Example

```
Script(Screen_Info)
String(StrMessage)
Number(nColumns)
Number(nRows)
Number(nPositionRow)
Number(nPositionColumn)
Activate(From_Menu)
    nRows = Get_Screen_Rows
    nColumns = Get_Screen_Columns
    nPositionRow = Get_Position_Row
    nPositionColumn = Get_Position_Column
    strMessage = String_Combine("Screen:",
    Number_To_String_Decimal(nRows))
    strMessage = String_Combine(strMessage, "rows,")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nColumns))
    strMessage = String_Combine(strMessage, "columns")
    Message(strMessage, 10)
    strMessage = String_Combine("Cursor position: row",
    Number_To_String_Decimal(nPositionRow))
    strMessage = String_Combine(strMessage, " column")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nPositionColumn))
    Ask_OK(strMessage, "Screen_Info")
Return
```

---

### See Also

Get\_Screen\_Columns, Get\_Screen\_Rows, Get\_Position\_Column, Set\_Cursor\_Position,  
Get\_Field\_Row, Get\_Field\_Column

## Get\_Session\_Number

Get the number for the session in which this script is executing.

---

### Example

```
Script(Get_Session_Number_Test)
String(strMessage)
String(strSessionNumber)
Number(nSession)
Activate(From_Menu)
    nSession = Get_Session_Number
    strSessionNumber = Number_To_String_Decimal(nSession)
    strMessage = String_Combine("Session Number:",
    strSessionNumber)
    Message(strMessage, 7)
Return
```

---

### See Also

[Get\\_MAC\\_Address](#), [Get\\_IP\\_Address](#), [Get\\_Workstation\\_ID](#)

---

## Get\_Time

Get the amount of time passed since January 1, 2000.

---

### Return Value

Returns the number of seconds that have elapsed since January 1, 2000.

---

### Example

```
Script( Get_Time_Test )
Number( nStartTimeSeconds )
Number( nDelayMilliseconds )
Number( nDelaySeconds )
Number( nEndTimeSeconds )
Activate( From_Menu )
    nStartTimeSeconds = Get_Time
    nDelaySeconds = 2
    nDelayMilliseconds = Number_Multiply( nDelaySeconds,
    1000 )
    Message( String_Combine( Number_To_String_Decimal(
    nDelaySeconds ), " second delay..." ), nDelaySeconds )
    Delay( 2000 )
    nEndTimeSeconds = Get_Time
    nDelaySeconds = Number_Minus( nEndTimeSeconds,
    nStartTimeSeconds )
    Message( String_Combine( "The delay, in seconds, was:
    ", Number_To_String_Decimal( nDelaySeconds ) ), 10 )
Return
```

---

### See Also

[Delay](#), [Get\\_Time\\_Since\\_Reset](#), [Wait\\_For\\_Screen\\_Update\\_With\\_Timeout](#)

## Get\_Time\_Since\_Reset

Gets the amount of time since the last reboot.

---

### Return Value

Returns the number of milliseconds that the computer has been non-suspended since the last reboot.

---

### Example

```
Script( Get_Time_Since_Reset_Test )
String( strTitle )
String( strMessage )
Number( nMilliseconds )
Number( nSeconds )
Number( nMinutes )
Number( nHours )
Number( nDays )
Activate( From_Menu )
    strTitle = "The time since the last reboot, excluding time the device
was suspended"
    nMilliseconds = Get_Time_Since_Reset
    nSeconds = Number_Divide( nMilliseconds, 1000 )
    nMinutes = Number_Divide( nSeconds, 60 )
    nHours = Number_Divide( nMinutes, 60 )
    nDays = Number_Divide( nHours, 24 )
    strMessage = String_Combine( Number_To_String_Decimal( nMilliseconds
),
" milliseconds = " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nSeconds ) )
    strMessage = String_Combine( strMessage, " seconds = " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nMinutes ) )
    strMessage = String_Combine( strMessage, " minutes = " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nHours ) )
    strMessage = String_Combine( strMessage, " hours = " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nDays
```

```
) )  
    strMessage = String_Combine( strMessage, " days" )  
    Ask_OK( strMessage, strTitle )  
    Return
```

---

**See Also**

[Delay](#), [Wait\\_For\\_Screen\\_Update\\_With\\_Timeout](#), [Get\\_Time](#), [Reboot](#)

## Get\_Field\_Index

Get the index of a field at the specified row and column.

---

### Parameters

|               |                               |
|---------------|-------------------------------|
| <i>Row</i>    | The row containing the field. |
| <i>Column</i> | A column in the field.        |

---

### Format

`Get_Field_Index (Row, Column)`

---

### Remarks

An index of -1 means there is no field at the row and column. This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script( Get_Field_Index_Test )
String( strMessage )
Number( nRow )
Number( nColumn )
Number( nFieldIndex )
Activate( From_Menu )
    nRow = Ask_Number( "Enter the row number containing the field",
"Get_Field_Index", 1, 999, 1 )
    nColumn = Ask_Number( "Enter a column number in the field",
"Get_Field_Index", 1, 999, 1 )
    nFieldIndex = Get_Field_Index( nRow, nColumn )
    strMessage = String_Combine( "Field at row ",
Number_To_String_Decimal(
nRow ) )
    strMessage = String_Combine( strMessage, ", column " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
```

```
nColumn ) )
    strMessage = String_Combine( strMessage, ": " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nFieldIndex ) )
    Message( strMessage, 12 )
    Return
```

---

## See Also

[Get\\_Screen\\_Columns](#), [Get\\_Screen\\_Rows](#), [Get\\_Position\\_Column](#), [Get\\_Position\\_Row](#),  
[Get\\_Session\\_Number](#), [Get\\_Time](#), [Get\\_Time\\_Since\\_Reset](#), [Get\\_Num\\_Fields](#),  
[Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#), [Get\\_Field\\_Row](#),  
[Get\\_Field\\_Column](#), [Get\\_Field\\_Length](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#)

## Get\_Num\_Fields

Get the number of fields on the screen.

---

### Remarks

This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Row_Column_Length)
String(strMessage)
Number(numFields)
Number(nLoops)
Number(nFieldRow)
Number(nFieldColumn)
Number(nFieldLength)
Activate(From_Menu)
    numFields = Get_Num_Fields
    Message(String_Combine("Number of fields:",
    Number_To_String_Decimal(numFields)), 60)
    nLoops = 0
    While(Number_Less_Than(nLoops, numFields))
        nFieldRow = Get_Field_Row(nLoops)
        nFieldColumn = Get_Field_Column(nLoops)
        nFieldLength = Get_Field_Length(nLoops)
        strMessage = String_Combine("Field:",
        Number_To_String_Decimal(nLoops))
        strMessage = String_Combine(strMessage, ":row")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldRow))
        strMessage = String_Combine(strMessage, ", column")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldColumn))
        strMessage = String_Combine(strMessage, ", length")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldLength))
        Ask_OK(strMessage, "Field Info")
        nLoops = Number_Plus(nLoops, 1)
    End_While
    Comment: The following should return zero because the field
```

```
index is invalid.  
nFieldRow = Get_Field_Row(nLoops)  
nFieldColumn = Get_Field_Column(nLoops)  
nFieldLength = Get_Field_Length(nLoops)  
Return
```

---

## See Also

[Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#), [Get\\_Field\\_Row](#),  
[Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

## Get\_Field\_Index\_Row\_Text

Get the index of a field that is in the same row as the text.

---

### Parameters

|                          |  |
|--------------------------|--|
| <i>Screen Text</i>       | The text on the same row as the field.                                 |
| <i>Text Before Field</i> | Indicates whether the text is before or after the field.               |
| <i>Ignore Case</i>       | Indicates whether the case of the letters is taken into consideration. |

---

### Format

```
Get_Field_Index_Row_Text (Screen Text, Text Before Field, Ignore Case)
```

---

### Return Value

Returns a Boolean. TRUE if the text is before the field, FALSE if the text is after the field.

---

### Remarks

The text may be before or after the field in the same row as the field. An index of -1 means either the text was not found or there is no field before or after the text in the row where the text was found. This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Index_Row_Text_Test)  
String(strTextInRow)  
Boolean(bTextBeforeField)  
Number(nFieldIndex)
```

---

```
Activate(From_Menu)
    strTextInRow = Ask_String("Enter some text on the same row as the
field", "Get_Field_Index_Row_Text", 1, 99, "")
    bTextBeforeField = Ask_Yes_No("Is the text before the field?",
"Get_Field_Index_Row_Text", FALSE)
    nFieldIndex = Get_Field_Index_Row_Text(strTextInRow,
bTextBeforeField, FALSE)
    Message(String_Combine("Field ID (0 is first field): ",
Number_To_String_Decimal(nFieldIndex)), 5)
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Column\\_Text](#), [Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#),  
[Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

## Get\_Field\_Index\_Column\_Text

Get the index of a field that is in the same column as the text.

---

### Parameters

|                         |  |
|-------------------------|--|
| <i>Screen Text</i>      | The text in the same column as the field.                              |
| <i>Text Above Field</i> | Indicates whether the text is above or below the field.                |
| <i>Ignore Case</i>      | Indicates whether the case of the letters is taken into consideration. |

---

### Format

`Get_Field_Index_Column_Text (Screen Text, Text Above Field, Ignore Case)`

---

### Return Value

Returns a Boolean. TRUE if the text is above the field, FALSE if the text is below the field.

---

### Remarks

The text may be above or below the field in the same row as the field. An index of -1 means either the text was not found or there is no field before or after the text in the column where the text was found. This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Index_Column_Text_Test)
String(strTextInColumn)
Boolean(bTextAboveField)
Number(nFieldIndex)
```

---

```
Activate(From_Menu)
    strTextInColumn = Ask_String("Enter some text in the same column as the
field", "Get_Field_Index_Column_Text", 1, 99, "")
    bTextAboveField = Ask_Yes_No("Is the text above the field?",
    "Get_Field_Index_Column_Text", FALSE)
    nFieldIndex = Get_Field_Index_Column_Text(strTextInColumn,
    bTextAboveField, FALSE)
    Message(String_Combine("Field ID (0 is first field): ",
    Number_To_String_Decimal(nFieldIndex)), 5)
    Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#),  
[Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

# Get\_Field\_Row

Get the row number of the field.

---

## Parameters

*Field Index*                      The index of the field.

---

## Format

Get\_Field\_Row (Field Index)

---

## Example

```
Script(Get_Field_Row_Column_Length)
String(strMessage)
Number(numFields)
Number(nLoops)
Number(nFieldRow)
Number(nFieldColumn)
Number(nFieldLength)
Activate(From_Menu)
    numFields = Get_Num_Fields
    Message(String_Combine("Number of fields:",
    Number_To_String_Decimal(numFields)), 60)
    nLoops = 0
    While(Number_Less_Than(nLoops, numFields))
        nFieldRow = Get_Field_Row(nLoops)
        nFieldColumn = Get_Field_Column(nLoops)
        nFieldLength = Get_Field_Length(nLoops)
        strMessage = String_Combine("Field:",
        Number_To_String_Decimal(nLoops))
        strMessage = String_Combine(strMessage, ":row")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldRow))
        strMessage = String_Combine(strMessage, ", column")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldColumn))
        strMessage = String_Combine(strMessage, ", length")
```

---

```
    strMessage = String_Combine(strMessage,  
    Number_To_String_Decimal(nFieldLength))  
    Ask_OK(strMessage, "Field Info")  
    nLoops = Number_Plus(nLoops, 1)  
End_While  
Comment: The following should return zero because the field  
index is invalid.  
nFieldRow = Get_Field_Row(nLoops)  
nFieldColumn = Get_Field_Column(nLoops)  
nFieldLength = Get_Field_Length(nLoops)  
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

# Get\_Field\_Column

Get the column number of the field.

---

## Parameters

*Field Index*                      The index of the field.

---

## Format

Get\_Field\_Column (Field Index)

---

## Example

```
Script(Get_Field_Row_Column_Length)
String(strMessage)
Number(numFields)
Number(nLoops)
Number(nFieldRow)
Number(nFieldColumn)
Number(nFieldLength)
Activate(From_Menu)
    numFields = Get_Num_Fields
    Message(String_Combine("Number of fields:",
    Number_To_String_Decimal(numFields)), 60)
    nLoops = 0
    While(Number_Less_Than(nLoops, numFields))
        nFieldRow = Get_Field_Row(nLoops)
        nFieldColumn = Get_Field_Column(nLoops)
        nFieldLength = Get_Field_Length(nLoops)
        strMessage = String_Combine("Field:",
        Number_To_String_Decimal(nLoops))
        strMessage = String_Combine(strMessage, ":row")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldRow))
        strMessage = String_Combine(strMessage, ", column")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldColumn))
        strMessage = String_Combine(strMessage, ", length")
```

---

```
    strMessage = String_Combine(strMessage,  
    Number_To_String_Decimal(nFieldLength))  
    Ask_OK(strMessage, "Field Info")  
    nLoops = Number_Plus(nLoops, 1)  
End_While  
Comment: The following should return zero because the field  
index is invalid.  
nFieldRow = Get_Field_Row(nLoops)  
nFieldColumn = Get_Field_Column(nLoops)  
nFieldLength = Get_Field  
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#),  
[Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_ID](#)

# Get\_Field\_Length

Get the length of the field.

---

## Parameters

*Field Index*                      The index of the field.

---

## Format

Get\_Field\_Length (Field Index)

---

## Example

```
Script(Get_Field_Row_Column_Length)
String(strMessage)
Number(numFields)
Number(nLoops)
Number(nFieldRow)
Number(nFieldColumn)
Number(nFieldLength)
Activate(From_Menu)
    numFields = Get_Num_Fields
    Message(String_Combine("Number of fields:",
    Number_To_String_Decimal(numFields)), 60)
    nLoops = 0
    While(Number_Less_Than(nLoops, numFields))
        nFieldRow = Get_Field_Row(nLoops)
        nFieldColumn = Get_Field_Column(nLoops)
        nFieldLength = Get_Field_Length(nLoops)
        strMessage = String_Combine("Field:",
        Number_To_String_Decimal(nLoops))
        strMessage = String_Combine(strMessage, ":row")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldRow))
        strMessage = String_Combine(strMessage, ", column")
        strMessage = String_Combine(strMessage,
        Number_To_String_Decimal(nFieldColumn))
        strMessage = String_Combine(strMessage, ", length"
```

---

```
    strMessage = String_Combine(strMessage,  
    Number_To_String_Decimal(nFieldLength))  
    Ask_OK(strMessage, "Field Info")  
    nLoops = Number_Plus(nLoops, 1)  
End_While  
Comment: The following should return zero because the field  
index is invalid.  
nFieldRow = Get_Field_Row(nLoops)  
nFieldColumn = Get_Field_Column(nLoops)  
nFieldLength = Get_Field_Length(nLoops)  
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Symbology\\_Operator](#), [Set\\_Field\\_Append\\_Scan\\_Data](#),  
[Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#), [Get\\_Field\\_Append\\_Scan\\_Data](#),  
[Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

## Get\_Num\_Field\_Data\_IDs

Get the number of data IDs in a field.

---

### Parameters

*Field Index*                      The index of the field.

---

### Format

Get\_Num\_Field\_Data\_IDs (Field Index)

---

### Remarks

The number -1 means that the field index is not valid.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

---

### Example

```
Script(Get_Field_Data_ID_Test)
String(strDataID)
Number(numDataIDs)
Number(counter)
Activate(From_Menu)
    numDataIDs = Get_Num_Field_Data_IDs(0)
    counter = 0
    While(Number_Less_Than(counter, numDataIDs))
        strDataID = Get_Field_Data_ID(0, counter)
        Ask_OK(strDataID, "Data ID for Field 0")
        counter = Number_Plus(counter, 1)
```

---

End\_While  
Return

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Symbology\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

## Get\_Num\_Field\_Symbology\_IDs

Get the number of symbology IDs in a field.

---

### Parameters

*Field Index*                      The index of the field.

---

### Format

Get\_Num\_Field\_Symbology\_IDs (Field Index)

---

### Remarks

The number `-1` means the field index is not valid.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---

### Example

```
Script(Get_Field_Symbology_ID_Test)
String(strSymbologyID)
Boolean(ok)
Number(numSymbologies)
Number(counter)
Activate(From_Menu)
    Comment: Set some symbologies for field 0, then display them.
    ok = Set_Field_Symbology_ID(0, "UPCE0", FALSE)
    ok = Set_Field_Symbology_ID(0, "CODE 39", FALSE)
    ok = Set_Field_Symbology_ID(0, "EAN8", FALSE)
    numSymbologies = Get_Num_Field_Symbology_IDs(0)
    counter = 0
    While(Number_Less_Than(counter, numSymbologies))
        strSymbologyID = Get_Field_Symbology_ID(0, counter)
        Ask_OK(strSymbologyID, "Symbology for Field 0")
```

---

```
    counter = Number_Plus(counter, 1)
End_While
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

## Get\_Field\_Com\_Data\_Field

Get the index of the field that is the Com Data Field.

---

### Remarks

An index of -1 means that no field is the Com Data Field.

---

**NOTE** This action is only valid when using IBM 5250 or 5555 emulation.

---



---

### Example

```
Script(SetGet_Field_Com_Data_Field_Test)
Boolean(bSetOK)
Number(nFieldID)
Activate(From_Menu)
  bSetOK = Set_Field_Com_Data_Field(2, TRUE)
  If(bSetOK)
    nFieldID = Get_Field_Com_Data_Field
    Message(String_Combine("Get_Field_Com_Data_Field: ",
      Number_To_String_Decimal(nFieldID)), 7)
  Else
    Message("Set_Field_Com_Data_Field failed", 5)
  End_If
Return
```

---

### See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Set\\_Field\\_Data\\_ID](#), [Set\\_Field\\_Symbology\\_ID](#),  
[Get\\_Field\\_Symbology\\_ID](#)

---

# Get\_Scan\_Type\_Value

Get the number value of the supplied scan type name.

---

## Parameters

*Scan Type Name*                      The name of the scan type.

---

## Format

Get\_Scan\_Type\_Value (Scan Type Name)

---

## Return Value

Returns the value of the supplied scan type name.

---

## Remarks

A value of 0 is returned if the scan type name is not recognized. Scanner symbology values can be found in *Symbologies and Values* on page 322.

---

## Example

```
Script( Get_Scan_Type_Value_Test )
String( strScanType )
String( strMessage )
Number( nScanNumberValue )
Activate( From_Menu )
    strScanType = Ask_String_Uppercase( "Enter the scan type, like
    ""UPCA"", "Get_Scan_Type_Value", 1, 99, "" )
    nScanNumberValue = Get_Scan_Type_Value( strScanType )
    strMessage = String_Combine( "Scan value for """, strScanType )
    strMessage = String_Combine( strMessage, """: " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nScanNumberValue ) )
```

```
Message( strMessage, 7 )  
Return
```

---

## See Also

[Get\\_Num\\_Fields](#), [Get\\_Field\\_Index\\_Row\\_Text](#), [Get\\_Field\\_Index\\_Column\\_Text](#),  
[Get\\_Field\\_Row](#), [Get\\_Field\\_Length](#), [Get\\_Field\\_Symbology\\_Operator](#),  
[Set\\_Field\\_Append\\_Scan\\_Data](#), [Set\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Prefix\\_Scan\\_Data](#),  
[Get\\_Field\\_Append\\_Scan\\_Data](#), [Get\\_Field\\_Column](#), [Get\\_Num\\_Field\\_Data\\_IDs](#),  
[Get\\_Num\\_Field\\_Symbology\\_IDs](#), [Get\\_Field\\_Com\\_Data\\_Field](#), [Set\\_Field\\_Data\\_ID](#),  
[Set\\_Field\\_Symbology\\_ID](#), [Get\\_Field\\_Symbology\\_ID](#)

---

## Ask\_Yes\_No\_Cancel

Displays a message in a dialog box with a **Yes**, **No**, and **Cancel** button and waits until the user selects a button.

---

### Parameters

|                        |  |
|------------------------|--|
| <i>Message Text</i>    | The message displayed in the message box.          |
| <i>Title Text</i>      | The title displayed on the message box.            |
| <i>Make No Default</i> | Indicates whether <b>No</b> is the default button. |

---

### Format

```
Ask_Yes_No_Cancel ("Message Text", "Title Text", Make No Default)
```

---

### Return Value

Returns 2 if the user presses **Yes**, 1 if the user presses **No**, and 0 if the user presses **Cancel**.

---

### Remarks

If the *Make No Default* value is TRUE, then the **No** button is the default. Otherwise, the **Yes** button is the default.

---

### Example

```
Script(Ask_Yes_No_Cancel_Test)
Number(nResult)
Activate(From_Menu)
    nResult = Ask_Yes_No_Cancel("Select Yes, No, or
    Cancel", "Ask_Yes_No_Cancel", FALSE)
    If(Number_Equal(nResult, 0))
        Message("Cancel", 3)
```

```
Else
  If(Number_Equal(nResult, 1))
    Message("No", 3)
  Else
    Message("Yes", 3)
  End_If
End_If
Return
```

---

## **See Also**

[Ask\\_Number](#), [Ask\\_Ok](#), [Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#), [Ask\\_String](#),  
[Ask\\_String\\_Password](#), [Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [Message](#)

---

# Run\_Application

Starts an application with the flags (optional) or a known type file with no flags.

---

## Parameters

|               |  |
|---------------|--|
| <i>Value1</i> | The path to the application (a string).      |
| <i>Value2</i> | The flags to run the application (a string). |
| <i>Value3</i> | Boolean (Wait Until Exit).                   |

---

## Format

```
Run_Application_Test ("String", "String", Boolean)
```

---

## Returns

If the application fails to start, a value of -1 will be returned.

If the `\ "Wait Until Exit\"` value is TRUE, the value returned will be the exit code for the application. Otherwise, a value of 0 will be returned.

---

## Remarks

Specifying the full path to the application or file is recommended.

---

## Example

```
Script( Run_Application_Test )
String( strExit )
Number( nExitCode )
Activate( From_Menu )
    nExitCode = Run_Application( "Notepad.exe", "", TRUE )
    strExit = Number_To_String_Decimal( nExitCode )
```

```
Ask_OK( strExit, "Exit Code" )  
Return
```



## String\_Find\_First

Finds the first instance of the substring inside the string, and returns the position where that substring starts.

---

### Parameters

|                          |  |
|--------------------------|--|
| <i>String to Parse</i>   | The string that gets searched.   |
| <i>Substring to Find</i> | The instance of the substring to find in the parsed string.            |
| <i>Ignore Case</i>       | Indicates whether the case of the letters is taken into consideration. |

---

### Format

`String_Find_First (String to Parse, Substring to Find, Ignore Case)`

---

### Remarks

The left-most position is 0, so a value of 0 would be returned if the string started with the substring. A value of -1 is returned if no instances of the substring are in the string.

---

### Example

```
Script( String_Find )
String( strOriginal )
String( strMessage )
String( strTitle )
String( strSearchFor )
Number( nFirstIndex )
Number( nLastIndex )
Activate( From_Menu )
    strOriginal = Ask_String( "Enter a string to search", "String_Find",
1,
99, "abcdef ABCDE" )
```

---

```
strSearchFor = Ask_String( "Search for:", "String_Find", 1, 99, "bc" )
nFirstIndex = String_Find_First( strOriginal, strSearchFor, TRUE )
nLastIndex = String_Find_Last( strOriginal, strSearchFor, TRUE )
strTitle = String_Combine( "Search """, strOriginal )
strTitle = String_Combine( strTitle, "" for "" )
strTitle = String_Combine( strTitle, strSearchFor )
strTitle = String_Combine( strTitle, """" )
strMessage = String_Combine( "String_Find_First: ",
Number_To_String_Decimal( nFirstIndex ) )
Ask_OK( strMessage, strTitle )
strMessage = String_Combine( "String_Find_Last: ",
Number_To_String_Decimal( nLastIndex ) )
Ask_OK( strMessage, strTitle )
Return
```

---

## See Also

[String\\_Length](#), [String\\_Find\\_Last](#), [String\\_Equal](#), [String\\_Set](#)

## String\_Find\_Last

Finds the last instance of the substring inside the string, and returns the position where that substring starts.

---

### Parameters

|                          |  |
|--------------------------|--|
| <i>String to Parse</i>   | The string that gets searched.   |
| <i>Substring to Find</i> | The instance of the substring to find in the parsed string.            |
| <i>Ignore Case</i>       | Indicates whether the case of the letters is taken into consideration. |

---

### Format

`String_Find_Last (String to Parse, Substring to Find, Ignore Case)`

---

### Remarks

The left-most position is 0, so a value of 0 would be returned if the string started with the substring. A value of -1 is returned if no instances of the substring are in the string.

---

### Example

```
Script( String_Find )
String( strOriginal )
String( strMessage )
String( strTitle )
String( strSearchFor )
Number( nFirstIndex )
Number( nLastIndex )
Activate( From_Menu )
    strOriginal = Ask_String( "Enter a string to search", "String_Find",
1,
99, "abcdef ABCDE" )
```

---

```
strSearchFor = Ask_String( "Search for:", "String_Find", 1, 99, "bc" )
nFirstIndex = String_Find_First( strOriginal, strSearchFor, TRUE )
nLastIndex = String_Find_Last( strOriginal, strSearchFor, TRUE )
strTitle = String_Combine( "Search """, strOriginal )
strTitle = String_Combine( strTitle, "" for "" )
strTitle = String_Combine( strTitle, strSearchFor )
strTitle = String_Combine( strTitle, """" )
strMessage = String_Combine( "String_Find_First: ",
Number_To_String_Decimal( nFirstIndex ) )
Ask_OK( strMessage, strTitle )
strMessage = String_Combine( "String_Find_Last: ",
Number_To_String_Decimal( nLastIndex ) )
Ask_OK( strMessage, strTitle )
Return
```

---

## See Also

[String\\_Length](#), [String\\_Find\\_First](#), [String\\_Equal](#), [String\\_Set](#)



---

```
"Number_Actions", 0, 2147483647, 36)
nNumber2 = Ask_Number("Enter the second number",
"Number_Actions", 0, 2147483647, 5)
nSum = Number_Plus(nNumber1, nNumber2)
nDifference = Number_Minus(nNumber1, nNumber2)
nProduct = Number_Multiply(nNumber1, nNumber2)
nQuotient = Number_Divide(nNumber1, nNumber2)
nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
strNumbers =
String_Combine(Number_To_String_Decimal(nNumber1), ", ")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number_Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number_Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number_Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number_Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number_Divide_Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Plus](#), [Number\\_Minus](#), [Number\\_Multiply](#), [Number\\_Divide](#),  
[Number\\_Divide\\_Remainder](#), [Ask\\_Number](#), [Number\\_Equal](#)

# Number\_Plus

Add two numbers together and return the sum.

---

## Parameters

|                |                  |
|----------------|------------------|
| <i>Number1</i> | The first term.  |
| <i>Number2</i> | The second term. |

---

## Format

```
Number_Plus (Number1, Number2)
```

---

## Remarks

Each parameter may be a constant or a variable or an action.

---

## Example

```
Script(Number_Actions)
String(strMessage)
String(strTitle)
String(strNumbers)
Number(nNumber1)
Number(nNumber2)
Number(nSum)
Number(nDifference)
Number(nProduct)
Number(nQuotient)
Number(nRemainder)
Activate(From_Menu)
    nNumber1 = Ask_Number("Enter the first number",
        "Number_Actions", 0, 2147483647, 36)
    nNumber2 = Ask_Number("Enter the second number",
        "Number_Actions", 0, 2147483647, 5)
    nSum = Number_Plus(nNumber1, nNumber2)
```

```
nDifference = Number_Minus(nNumber1, nNumber2)
nProduct = Number_Multiply(nNumber1, nNumber2)
nQuotient = Number_Divide(nNumber1, nNumber2)
nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
strNumbers =
String_Combine(Number_To_String_Decimal(nNumber1), ",")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number Divide_Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Set](#), [Number\\_Minus](#), [Number\\_Multiply](#), [Number\\_Divide](#),  
[Number\\_Divide\\_Remainder](#), [Ask\\_Number](#), [Number\\_Equal](#)

# Number\_Minus

Subtract the second term from the first term to get the difference.

---

## Parameters

*Number1*                                      The first term.

*Number2*                                      The second term.

---

## Format

Number\_Minus (Number1, Number2)

---

## Return Value

Returns the value when Number2 is subtracted from Number1.

---

## Example

```
Script(Number_Actions)
String(strMessage)
String(strTitle)
String(strNumbers)
Number(nNumber1)
Number(nNumber2)
Number(nSum)
Number(nDifference)
Number(nProduct)
Number(nQuotient)
Number(nRemainder)
Activate(From_Menu)
    nNumber1 = Ask_Number("Enter the first number",
        "Number_Actions", 0, 2147483647, 36)
    nNumber2 = Ask_Number("Enter the second number",
        "Number_Actions", 0, 2147483647, 5)
    nSum = Number_Plus(nNumber1, nNumber2)
```

---

```
nDifference = Number_Minus(nNumber1, nNumber2)
nProduct = Number_Multiply(nNumber, nNumber2)
nQuotient = Number_Divide(nNumber1, nNumber2)
nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
strNumbers =
String_Combine(Number_To_String_Decimal(nNumber1), ", ")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number Divide_Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Set](#), [Number\\_Plus](#), [Number\\_Multiply](#), [Number\\_Divide](#),  
[Number\\_Divide\\_Remainder](#), [Ask\\_Number](#), [Number\\_Equal](#)

# Number\_Multiply

Multiply the first term by the second term and returns the product.

---

## Parameters

|                |                  |
|----------------|------------------|
| <i>Number1</i> | The first term.  |
| <i>Number2</i> | The second term. |

---

## Format

```
Number_Multiply (Number1, Number2)
```

---

## Remarks

Each parameter may be a constant, variable, or action.

---

## Example

```
Script(Number_Actions)
String(strMessage)
String(strTitle)
String(strNumbers)
Number(nNumber1)
Number(nNumber2)
Number(nSum)
Number(nDifference)
Number(nProduct)
Number(nQuotient)
Number(nRemainder)
Activate(From_Menu)
    nNumber1 = Ask_Number("Enter the first number",
        "Number_Actions", 0, 2147483647, 36)
    nNumber2 = Ask_Number("Enter the second number",
        "Number_Actions", 0, 2147483647, 5)
    nSum = Number_Plus(nNumber1, nNumber2)
```

---

```
nDifference = Number_Minus(nNumber1, nNumber2)
nProduct = Number_Multiply(nNumber1, nNumber2)
nQuotient = Number_Divide(nNumber1, nNumber2)
nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
strNumbers =
String_Combine(Number_To_String_Decimal(nNumber1), ",")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number Divide Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Set](#), [Number\\_Plus](#), [Number\\_Minus](#), [Number\\_Divide](#),  
[Number\\_Divide\\_Remainder](#), [Ask\\_Number](#), [Number\\_Equal](#)



---

```
nSum = Number_Plus(nNumber1, nNumber2)
nDifference = Number_Minus(nNumber1, nNumber2)
nProduct = Number_Multiply(nNumber1, nNumber2)
nQuotient = Number_Divide(nNumber1, nNumber2)
nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
strNumbers =
String_Combine(Number_To_String_Decimal(nNumber1), ",")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number_Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number_Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number_Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number_Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number_Divide_Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Set](#), [Number\\_Plus](#), [Number\\_Minus](#), [Number\\_Multiply](#),  
[Number\\_Divide\\_Remainder](#), [Ask\\_Number](#), [Number\\_Equal](#)

## Number\_Divide\_Remainder

Divide the first term by the second term and return the remainder. For example, 7 divided by 3 would return a remainder of 1.

---

### Parameters

|                |                  |
|----------------|------------------|
| <i>Number1</i> | The first term.  |
| <i>Number2</i> | The second term. |

---

### Format

`Number_Divide_Remainder (Number1, Number2)`

---

### Example

```
Script(Number_Actions)
String(strMessage)
String(strTitle)
String(strNumbers)
Number(nNumber1)
Number(nNumber2)
Number(nSum)
Number(nDifference)
Number(nProduct)
Number(nQuotient)
Number(nRemainder)
Activate(From_Menu)
    nNumber1 = Ask_Number("Enter the first number",
        "Number_Actions", 0, 2147483647, 36)
    nNumber2 = Ask_Number("Enter the second number",
        "Number_Actions", 0, 2147483647, 5)
    nSum = Number_Plus(nNumber1, nNumber2)
    nDifference = Number_Minus(nNumber1, nNumber2)
    nProduct = Number_Multiply(nNumber1, nNumber2)
    nQuotient = Number_Divide(nNumber1, nNumber2)
    nRemainder = Number_Divide_Remainder(nNumber1, nNumber2)
    strNumbers =
```

---

```
String_Combine(Number_To_String_Decimal(nNumber1), ",")
strNumbers = String_Combine(strNumbers,
Number_To_String_Decimal(nNumber2))
strTitle = String_Combine("Number_Plus", strNumbers)
Ask_OK(Number_To_String_Decimal(nSum), strTitle)
strTitle = String_Combine("Number_Minus", strNumbers)
Ask_OK(Number_To_String_Decimal(nDifference), strTitle)
strTitle = String_Combine("Number_Multiply", strNumbers)
Ask_OK(Number_To_String_Decimal(nProduct), strTitle)
strTitle = String_Combine("Number_Divide", strNumbers)
Ask_OK(Number_To_String_Decimal(nQuotient), strTitle)
strTitle = String_Combine("Number_Divide_Remainder",
strNumbers)
Ask_OK(Number_To_String_Decimal(nRemainder), strTitle)
Return
```

---

## See Also

[Number\\_Set](#), [Number\\_Plus](#), [Number\\_Minus](#), [Number\\_Multiply](#), [Number\\_Divide](#),  
[Ask\\_Number](#), [Number\\_Equal](#)



---

```
strBinary = strEntered
strDecimal = Number_To_String_Decimal( nBinary )
strHexUpper = Number_To_String_Hexadecimal_Uppercase( nBinary )
strOctal = Number_To_String_Octal( nBinary )
strMessage = String_Combine( "Binary:", strBinary )
strMessage = String_Combine( strMessage, "; Octal: " )
strMessage = String_Combine( strMessage, strOctal )
strMessage = String_Combine( strMessage, "; Decimal: " )
strMessage = String_Combine( strMessage, strDecimal )
strMessage = String_Combine( strMessage, ";
Hexadecimal:" )
strMessage = String_Combine( strMessage, strHexUpper )
Ask_OK( strMessage, "String_To_Number_Binary" )
Return
```

---

## See Also

[String\\_To\\_Number\\_Octal](#), [String\\_To\\_Number\\_Decimal](#),  
[String\\_To\\_Number\\_Hexadecimal](#), [Number\\_Equal](#), [String\\_Set](#), [Number\\_Set](#)



```
strOctal = strEntered
strBinary = Number_To_String_Binary(nOctal)
strDecimal = Number_To_String_Decimal(nOctal)
strHexUpper = Number_To_String_Hexadecimal_Uppercase(nOctal)
strMessage = String_Combine("Binary:", strBinary)
strMessage = String_Combine(strMessage, "; Octal:")
strMessage = String_Combine(strMessage, strOctal)
strMessage = String_Combine(strMessage, "; Decimal:")
strMessage = String_Combine(strMessage, strDecimal)
strMessage = String_Combine(strMessage, "; Hexadecimal:")
strMessage = String_Combine(strMessage, strHexUpper)
Ask_OK(strMessage, "String_To_Number_Octal")
Return
```

---

## See Also

[String\\_To\\_Number\\_Binary](#), [String\\_To\\_Number\\_Decimal](#),  
[String\\_To\\_Number\\_Hexadecimal](#), [Number\\_Equal](#), [String\\_Set](#), [Number\\_Set](#)



---

```
strDecimal = strEntered
strBinary = Number_To_String_Binary(nDecimal)
strHexUpper =
Number_To_String_Hexadecimal_Uppercase(nDecimal)
strOctal = Number_To_String_Octal(nDecimal)
strMessage = String_Combine("Binary:", strBinary)
strMessage = String_Combine(strMessage, "; Octal:")
strMessage = String_Combine(strMessage, strOctal)
strMessage = String_Combine(strMessage, "; Decimal:")
strMessage = String_Combine(strMessage, strDecimal)
strMessage = String_Combine(strMessage, "; Hexadecimal:")
strMessage = String_Combine(strMessage, strHexUpper)
Ask_OK(strMessage, "String_To_Number_Decimal")
Return
```

---

## See Also

[String\\_To\\_Number\\_Binary](#), [String\\_To\\_Number\\_Octal](#),  
[String\\_To\\_Number\\_Hexadecimal](#), [Number\\_Equal](#), [String\\_Set](#), [Number\\_Set](#)



```
strHexUpper = strEntered
strBinary = Number_To_String_Binary(nHexadecimal)
strDecimal = Number_To_String_Decimal(nHexadecimal)
strOctal = Number_To_String_Octal(nHexadecimal)
strMessage = String_Combine("Binary:", strBinary)
strMessage = String_Combine(strMessage, "; Octal:")
strMessage = String_Combine(strMessage, strOctal)
strMessage = String_Combine(strMessage, "; Decimal:")
strMessage = String_Combine(strMessage, strDecimal)
strMessage = String_Combine(strMessage, "; Hexadecimal:")
strMessage = String_Combine(strMessage, strHexUpper)
Ask_OK(strMessage, "String_To_Number_Hexadecimal")
Return
```

---

## See Also

[String\\_To\\_Number\\_Binary](#), [String\\_To\\_Number\\_Octal](#), [String\\_To\\_Number\\_Decimal](#), [Number\\_Equal](#), [String\\_Set](#), [Number\\_Set](#)

# Ask\_Number

Displays a dialog box asking the user for a decimal number.

---

## Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <i>Message Text</i>  | The message displayed in the box.     |
| <i>Title Text</i>    | The title of the message box.         |
| <i>Minimum Value</i> | The smallest value of the number.     |
| <i>Maximum Value</i> | The largest value of the number.      |
| <i>Default Value</i> | The initial value in the message box. |

---

## Format

```
Ask_Number ("Message Text", "Title Text", Minimum Value, Maximum Value,  
Default Value)
```

---

## Return Value

Returns the number supplied by the user.

---

## Remarks

The supplied default value is returned if the user cancels the dialog.

---

## Example

```
Script(Number_Convert)  
String(strEntered)  
String(strBinary)  
String(strHexLower)
```

---

```
String(strHexUpper)
String(strOctal)
Number(numEntered)
Activate(From_Menu)
    numEntered = Ask_Number("Enter the decimal number to convert",
        "Number_Convert", -22, 2000000000, 31)
    strEntered = Number_To_String_Decimal(numEntered)
    strBinary = Number_To_String_Binary(numEntered)
    strHexLower =
        Number_To_String_Hexadecimal_Lowercase(numEntered)
    strHexUpper =
        Number_To_String_Hexadecimal_Uppercase(numEntered)
    strOctal = Number_To_String_Octal(numEntered)
    Ask_OK(strBinary, String_Combine("Binary value of ",
        strEntered))
    Ask_OK(strHexLower, String_Combine("Hex (lower case) value of ",
        strEntered))
    Ask_OK(strHexUpper, String_Combine("Hex (upper case) value of ",
        strEntered))
    Ask_OK(strOctal, String_Combine("Octal value of ",
        strEntered))
    Return
```

---

## See Also

[Ask\\_Ok](#), [Ask\\_OK\\_Cancel](#), [Ask\\_Yes\\_No](#), [Ask\\_String](#), [Ask\\_String\\_Password](#),  
[Ask\\_String\\_Uppercase](#), [Ask\\_String\\_Lowercase](#), [Ask\\_Yes\\_No\\_Cancel](#), [Message](#)

# Character\_To\_Number

Converts the character at position Index in the string into the number value for that character.

---

## Parameters

|               |   |
|---------------|---|
| <i>String</i> | The string containing the conversion character. |
| <i>Index</i>  | The index of the character in the string.       |

---

## Format

Character\_To\_Number (String, Index)

---

## Return Value

Returns a character's number value. If the index does not point to a character, a value of 0 is returned.

---

## Remarks

An index of 0 indicates the left-most character in the string.

---

## Example

```
Script(Character_To_Number_Test)
String(strCharacters)
String(strTitle)
Number(nIndex)
Number(nCharacter)
Number(nToConvert)
Activate(From_Menu)
    strCharacters = Ask_String("Enter a string",
```

```
"Character_To_Number", 1, 99, "abcde")
nIndex = Ask_Number("Enter the index of the character to
convert to a number", "Character_To_Number", 0, 99, 0)
nCharacter = Character_To_Number(strCharacters, nIndex)
strTitle = String_Combine("","", strTitle)
strTitle =
String_Combine(Number_To_Character(nCharacter),
strTitle)
strTitle = String_Combine("Character_To_Number of","",
strTitle)
Ask_OK(Number_To_String_Decimal(nCharacter), strTitle)
Return
```

---

## See Also

[Ask\\_String](#), [Search\\_Screen](#), [Speech\\_To\\_Text](#), [Get\\_Screen\\_Text](#)

## Bitwise\_And

The resulting number will have a bit set when both input numbers have that bit set.

---

### Parameters

*Value1*                                      Number; Number 1

*Value2*                                      Number; Number 2

---

### Format

Bitwise\_And (Value1, Value2)

---

### Example

## Bitwise\_Or

The resulting number will have a bit set when either input numbers has that bit set (inclusive or).

---

### Parameters

*Value1*                                      Number; Number 1

*Value2*                                      Number; Number 2

---

### Format

Bitwise\_Or (Value1, Value2)

---

### Example

## Bitwise\_Xor

The resulting number will have a bit set when exactly one input number has that bit set (exclusive or).

---

### Parameters

*Value1*                                      Number; Number 1

*Value2*                                      Number; Number 2

---

### Format

Bitwise\_Xor (Value1, Value2)

---

### Example



## Symbologies and Values

The following is a list of symbologies and their values:

| Symbology      | Value |
|----------------|-------|
| UPCE0          | 48    |
| UPCE1          | 49    |
| UPCA           | 50    |
| MSI            | 51    |
| EAN8           | 52    |
| EAN13          | 53    |
| CODABAR        | 54    |
| CODE 39        | 55    |
| D 2 OF 5       | 56    |
| I 2 OF 5       | 57    |
| CODE 11        | 58    |
| CODE 93        | 59    |
| CODE 128       | 60    |
| D 2 OF 5 IATA  | 62    |
| EAN/UCC 128    | 63    |
| PDF417         | 64    |
| TRIOPTIC 39    | 66    |
| COUPON CODE    | 67    |
| BOOKLAND       | 68    |
| MICROPDF       | 69    |
| CODE 32        | 70    |
| MACRO PDF      | 71    |
| MAXOCODE       | 72    |
| DATAMATRIX     | 73    |
| QR CODE        | 74    |
| MACRO MICROPDF | 75    |
| RSS 14         | 76    |
| RSS LIMITED    | 77    |
| RSS EXPANDED   | 78    |
| SIGNATURE      | 82    |
| WEBCODE        | 84    |

---

| Symbology        | Value |
|------------------|-------|
| CUECODE          | 85    |
| COMPOSITE        | 86    |
| TLC 39           | 88    |
| POSTNET          | 97    |
| PLANET           | 98    |
| BRITISH POSTAL   | 99    |
| JAPAN POSTAL     | 100   |
| AUSTRALIA POSTAL | 101   |
| DUTCH POSTAL     | 102   |
| CANADA POSTAL    | 103   |
| AZTEC            | 160   |
| AZTEC MESA       | 161   |
| CODE 49          | 162   |
| OCR              | 163   |
| CODABLOCK        | 164   |
| MATRIX 2 OF 5    | 165   |
| PLESSEY          | 166   |
| CHINA POSTAL     | 167   |
| KOREA POSTAL     | 168   |
| TELEPEN          | 169   |
| CODE 16K         | 170   |
| POSI CODE        | 171   |
| UPC              | 241   |
| MSR              | 245   |
| RFID             | 246   |

## Voice-Enabled Emulation Settings

This section lists the settings supported by Voice-Enabled Emulation. These settings are to be used in conjunction with the preceding scripting actions. The following settings are listed:

- Text-to-Speech Settings
- Speech-to-Text Settings

### Text-to-Speech Settings

| Action                                       | Description  |
|--|--|
| <a href="#">tts_language</a>                 | Displays the full name of the language currently being used.                 |
| <a href="#">tts_language_short</a>           | Displays the three-letter abbreviation of the language currently being used. |
| <a href="#">tts_language_long</a>            | Displays the full name of the language currently being used.                 |
| <a href="#">tts_voice</a>                    | Indicates the name of the voice that is currently selected.                  |
| <a href="#">tts_frequency</a>                | Indicates the sampling frequency.  |
| <a href="#">tts_volume</a>                   | Indicates the sound level.   |
| <a href="#">tts_rate</a>                     | Indicates the speed level.   |
| <a href="#">tts_readmode</a>                 | Indicates how text should be separated.                                      |
| <a href="#">tts_waitfactor</a>               | Indicates the length of the pause between messages.                          |
| <a href="#">tts_calibrate</a>                | Opens the speaker volume calibration wizard.                                 |
| <a href="#">tts_external_speaker_setting</a> | Speaker setting for use on Motorola/Symbol mobile devices.                   |

### Speech-to-Text Settings

| Action                             | Description  |
|------------------------------------|--|
| <a href="#">stt_domain</a>         | Indicates the situation in which speech-to-text is being used.               |
| <a href="#">stt_language</a>       | Displays the three-letter abbreviation of the language currently being used. |
| <a href="#">stt_language_short</a> | Displays the three-letter abbreviation of the language currently being used. |

| Action                               | Description   |
|--------------------------------------|---|
| <a href="#">stt_langue_long</a>      | Displays the full name of the language currently being used.  |
| <a href="#">stt_frequency</a>        | Displays the sampling frequency.  |
| <a href="#">stt_size</a>             | Displays the size of the speech-to-text engine being used.  |
| <a href="#">stt_timeout</a>          | Indicates the total milliseconds (ms) for the system to wait before responding to the speaker.  |
| <a href="#">stt_idle_timeout</a>     | Indicates the total milliseconds for the engine to continue collecting results following the last result or timeout.  |
| <a href="#">stt_silence</a>          | Indicates milliseconds of silence used to indicate the user is done speaking.   |
| <a href="#">stt_fx_silence</a>       | Indicates the milliseconds of silence used to indicate the user is done speaking.   |
| <a href="#">stt_expanded</a>         |   |
| <a href="#">stt_confidence</a>       | Indicates the minimum amount of difference between the confidence for the most likely and next-most likely items that will be accepted.   |
| <a href="#">stt_fx_detect_start</a>  | Indicates the action the speech engine should take before attempting to determine what the user is saying.  |
| <a href="#">stt_threshold</a>        | Indicates the minimum amount of confidence for the most-likely result that will be accepted.  |
| <a href="#">stt_fx_threshold</a>     | Indicates the amount of energy the microphone input must have before the speech detection is activated.   |
| <a href="#">stt_save_threshold</a>   | Directs the speech engine to save the state if the result confidence is greater than the result confidence for <code>stt_threshold</code> and <code>stt_save_threshold</code> combined. |
| <a href="#">stt_fx_min_duration</a>  | Indicates the minimum duration (in ms) of speech before speech detection is activated.  |
| <a href="#">stt_fx_sensitivity</a>   | Indicates the speech detection sensitivity.   |
| <a href="#">stt_volume</a>           | Indicates the current volume of the microphone input.   |
| <a href="#">stt_calibrate</a>        | Opens the microphone calibration wizard.  |
| <a href="#">stt_grammar_optimize</a> | Indicates the action the speech engine should take when a grammar is loaded.  |

| <b>Action</b>                           | <b>Description</b>   |
|---|--|
| <a href="#">stt_grammar_phonetic</a>    | Indicates whether the grammar is allowed to contain information on phonetic transcription of words.                                |
| <a href="#">stt_grammar_nonterminal</a> | Indicates whether an error will be generated if a non-terminal is found in the definition section of any modifiable rule.          |
| <a href="#">stt_context_optimize</a>    | Indicates the action the speech engine should take when a grammar is loaded.   |
| <a href="#">stt_processing</a>          | Indicates the action the speech engine should take when returning a grammar result.  |
| <a href="#">stt_save_session_delay</a>  | Indicates the total milliseconds for the speech engine to wait before saving the next current state.                               |
| <a href="#">stt_reset_session_delay</a> | Indicates the total milliseconds for the speech engine to wait for a valid response before reverting back to the last saved state. |
| <a href="#">stt_special_sounds</a>      | Indicates how the speech engine should interpret special sounds.   |
| <a href="#">stt_fx_microphone</a>       | Tells the speech engine the distance between the user and the microphone.  |
| <a href="#">stt_priority</a>            | Determines how aggressively the microphone input is collected and speech analysis is performed.                                    |

## tts\_language

Displays the full name of the language currently being used. For example, American English instead of ENU.

---

### Example

```
Script(Speech_Languages_Voices_Test)
Number(nVoice)
Number(nLanguage)
Activate(From_Menu)
  If_Not(Speech_From_Text_Available)
    Message("Speech From Text Not Available", 3)
    Return
  End_If
  nLanguage = Speech_Find_Setting_Value("tts_language",
  "Mexican Spanish", FALSE)
  If(Number_Greater_Than_Or_Equal(nLanguage, 0)
    Speech_Change_Setting("tts_language", nLanguage)
  End_If
  nVoice = Speech_Find_Setting_Value("tts_voice", "Javier",
  FALSE)
  If(Number_Greater_Than_Or_Equal(nVoice, 0))
    Speech_Change_Setting("tts_voice", nVoice)
    Speech_From_Text("La voc de Javier esta disponible.", FALSE)
  End_If
  nVoice = Speech_Find_Setting_Value("tts_voice", "Paulina",
  FALSE)
  If(Number_Greater_Than_Or_Equal(nVoice, 0))
    Speech_Change_Setting("tts_voice", nVoice)
    Speech_From_Text("La voz de Paulina esta disponible.",
  FALSE)
  End_If
  nLanguage = Speech_Find_Setting_Value("tts_language",
  "American English", FALSE)
  If(Number_Greater_Than_Or_Equal(nLanguage, 0))
    Speech_Change_Setting("tts_language", nLanguage)
  End_If
  nVoice = Speech_Find_Setting_Value("tts_voice", "tom", FALSE)
  If(Number_Greater_Than_Or_Equal(nVoice, 0))
    Speech_Change_Setting("tts_voice", nVoice)
    Speech_From_Text("Tom's voice is available.", FALSE)
  End_If
```

```
nVoice = Speech_Find_Setting_Value("tts_voice", "samantha", FALSE)
If(Number_Greater_Than_Or_Equal(nVoice, 0))
    Speech_Change_Setting("tts_voice", nVoice)
    Speech_From_Text("Samantha's voice is available.", FALSE)
End_If
nVoice = Speech_Find_Setting_Value("tts_voice", "jill", FALSE)
If(Number_Greater_Than_Or_Equal(nVoice, 0))
    Speech_Change_Setting("tts_voice", nVoice)
    Speech_From_Text("Jill's voice is a available.", FALSE)
End_If
Message("Speech Voice Testing Completed.", 3)
Return
```

---

## See Also

[tts\\_language\\_short](#), [tts\\_language\\_long](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),  
[Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## tts\_language\_short

Displays the three-letter abbreviation of the language currently being used. For example, ENU instead of American English.

---

### Example

```
Script(Speech_From_Text_Settings_Language)
String(strDescription)
String(strSetting)
Number(nLanguage)
Activate(From_Menu)
  If_Not(Speech_From_Text_Available)
    Message("Speech From Text Not Available", 3)
    Return
  End_If
  strSetting = "tts_language_short"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  strSetting = "tts_language_long"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  strSetting = "tts_language"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  Return
```

---

### See Also

[tts\\_language](#), [tts\\_language\\_long](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## tts\_language\_long

Displays the full name of the language currently being used.

---

### Example

```
Script(Speech_From_Text_Settings_Language)
String(strDescription)
String(strSetting)
Number(nLanguage)
Activate(From_Menu)
  If_Not(Speech_From_Text_Available)
    Message("Speech From Text Not Available", 3)
    Return
  End_If
  strSetting = "tts_language_short"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  strSetting = "tts_language_long"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  strSetting = "tts_language"
  nLanguage = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nLanguage)
  Ask_OK(strDescription, strSetting)
  Return
```

---

### See Also

[tts\\_language](#), [tts\\_language\\_short](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## tts\_voice

Indicates the name of the voice that is currently selected.

---

### Example

```
Script(Jill_Voice)
Number(nVoice)
Activate(From_Menu)
  If_Not(Speech_From_Text_Available)
    Message("Speech From Text Not Available", 3)
    Return
  End_If
nVoice = Speech_Find_Setting_Value("tts_voice", "Jill",
FALSE)
If(Number_Greater_Than_Or_Equal(nVoice, 0))
  Speech_Change_Setting("tts_voice", nVoice)
  Speech_From_Text("Jill's voice is available.", TRUE)
Else
  Speech_From_Text("Jill's voice is not available.",
  TRUE)
End_If
Return
```

---

### See Also

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## tts\_frequency

Indicates the sampling frequency.

---

### Possible Values

11 KHz  
16KHz  
22KHz

---

### Example

```
Script( Speech_From_Text_Frequency )
String( strDescription )
String( strSetting )
String( strMessage )
Number( nFrequency )
Activate( From_Menu )
    If_Not( Speech_From_Text_Available )
        Message( "Speech From Text Not Available", 3 )
        Return
    End_If
    strSetting = "tts_frequency"
    nFrequency = Speech_Get_Setting( strSetting )
    strDescription = Speech_Get_Setting_Value_Desc( strSetting,
nFrequency )
    strMessage = String_Combine( "Sampling Frequency:", strDescription )
    strMessage = String_Combine( strMessage, "kilohertz; setting value:"
)
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nFrequency ) )
    Speech_From_Text( strMessage, FALSE )
    Return
```

---

### See Also

[stt\\_frequency](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## tts\_volume

Indicates the sound level.

---

### Possible Values

Any number from 50 (silent) to 400 (loudest). The default value is 100.

---

### Example

```
Script( Speech_From_Text_Volume )
String( strDescription )
String( strSetting )
String( strMessage )
String( strPrompt )
Boolean( bResult )
Number( nVolume )
Number( nSettingMax )
Activate( From_Menu )
    If_Not( Speech_From_Text_Available )
        Message( "Speech From Text Not Available", 3 )
        Return
    End_If

    strSetting = "tts_volume"
    nVolume = Speech_Get_Setting( strSetting )
    strDescription = Speech_Get_Setting_Value_Desc( strSetting, nVolume )

    strMessage = String_Combine( "Volume: ", strDescription )
    strMessage = String_Combine( strMessage, "; setting value: " )
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nVolume ) )
    Speech_From_Text( strMessage, FALSE )
    nSettingMax = Speech_Get_Setting_Max( strSetting )
    strPrompt = String_Combine( "From 0 to ", Number_To_String_Decimal(
nSettingMax ) )
    While( Number_Not_Equal( nVolume, 0 ) )
        nVolume = Ask_Number( strPrompt, "New volume, 0 to exit", 0,
nSettingMax, nVolume )
        If( Number_Not_Equal( nVolume, 0 ) )
            bResult = Speech_Change_Setting( strSetting, nVolume )
            If( bResult )
```

```
        strMessage = String_Combine( "The new volume level is ",
Number_To_String_Decimal( nVolume ) )
        Speech_From_Text( strMessage, FALSE )
    Else
        Message( "Setting volume failed", 3 )
        nVolume = 0
    End_If
End_If
End_While
Return
```

---

### **See Also**

[stt\\_volume](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## tts\_rate

Indicates how fast the text should be spoken.

---

### Possible Values

Any number from 50-400, with 100 being the default (and "normal" talking speed), 50 being 1/2 normal speed, and 400 being 4 times faster than normal speed.

---

### Example

```
Script(Speech_Rates)
Number(nRate)
Activate(From_Menu)
    nRate = Speech_Get_Setting("tts_rate")
    Message(String_Combine("Initial speech rate is ",
        Number_To_String_Decimal(nRate)), 5)
    Speech_From_Text("This is the current speed.", TRUE)
    Speech_Change_Setting("tts_rate", 10)
    Speech_From_Text("This is the slow text.", TRUE)
    Speech_Change_Setting("tts_rate", 99)
    Speech_From_Text("This is the fast text.", TRUE)
    Speech_Change_Setting("tts_rate_", nRate)
Return
```

---

### See Also

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## tts\_readmode

Indicates how text should be separated.

---

### Possible Values

Sentence  
Character  
Word  
Line

---

### Example

```
Script(Speech_From_Text_ReadMode)
String(strDescription)
String(strSetting)
String(strMessage)
Number(nReadMode)
Activate(From_Menu)
  If_Not(Speech_From_Text_Available)
    Message("Speech From Text Not Available", 3)
    Return
  End_If
  strSetting = "tts_readmode"
  nReadMode = Speech_Get_Setting(strSetting)
  strDescription =
  Speech_Get_Setting_Value_Desc(strSetting, nReadMode)
  strMessage = String_Combine("Read-mode:",
  strDescription)
  strMessage = String_Combine(strMessage, "; setting
  value:")
  strMessage = String_Combine(strMessage,
  Number_To_String_Decimal(nReadMode))
  Speech_From_Text(strMessage, FALSE)
  Return
```

---

### See Also

Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc, Speech\_Get\_Setting\_Max,  
Speech\_Find\_Setting\_Value, Speech\_To\_Text, Speech\_From\_Text,  
Speech\_Setting\_Available, Speech\_Change\_Setting

## tts\_waitfactor

Indicates the length of the pause between messages.

---

### Possible Values

0 milliseconds (ms)  
200 ms  
400 ms  
600 ms  
800 ms  
1000 ms  
1200 ms

---

### Example

```
Script( Speech_From_Text_WaitFactor )
String( strDescription )
String( strSetting )
String( strMessage )
Number( nWaitFactor )
Activate( From_Menu )
    If_Not( Speech_From_Text_Available )
        Message( "Speech From Text Not Available", 3 )
        Return
    End_If
    strSetting = "tts_waitfactor"
    nWaitFactor = Speech_Get_Setting( strSetting )
    strDescription = Speech_Get_Setting_Value_Desc( strSetting,
nWaitFactor
)
    strMessage = String_Combine( "Wait-factor:", strDescription )
    strMessage = String_Combine( strMessage, ";milliseconds; setting
value:"
)
    strMessage = String_Combine( strMessage, Number_To_String_Decimal(
nWaitFactor ) )
```

```
Speech_From_Text( strMessage, FALSE )  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## **tts\_calibrate**

Opens the speaker volume calibration wizard.

---

### **Example**

```
Script(Alt_F10_Speaker_Calibrate)
Activate(On_Key, 0x79, Alt)
  Comment: Pressing Alt-F10 displays the speaker-calibration dialog.
  Speech_Change_Setting("tts_calibrate", 1)
Return
```

---

### **See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## tts\_external\_speaker\_setting

Speaker setting for use on Motorola/Symbol mobile devices.

---

### Possible Values

1 (on)

0 (off)

The default value is 0 (off).

---

### Remarks

If set to 1, only the CE device master volume will be adjusted. If set to 0, the volume will apply to the headset volume.

This setting is ignored for mobile devices from manufacturers other than Motorola/Symbol.

---

### Example

```
Script( Use_External_Speaker )
Activate( From_Menu )
    Comment: Change the speaker setting so Symbol devices volume changes
    affect the external speaker.
    If_Not( Speech_Change_Setting( "tts_external_speaker", 1 ) )
        Ask_OK( "This setting is not supported. Update your Vocalizer
version.", "Error" )
        Return
    End_If
    If_Not( Number_Equal( Speech_Get_Setting( "tts_external_speaker" ), 1
)
)
        Ask_OK( "The setting change was not preserved!", "Error" )
        Return
    End_If
    Comment: Perform a calibration so the user can set the volume.
```

```
Speech_Change_Setting( "tts_calibrate", 0 )  
Return
```

---

**See Also**

[tts\\_volume](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_domain

Indicates the situation in which speech-to-text is being used.

---

### Possible Values

Car  
Mobile

---

### Example

```
Script(Speech_To_Text_Setting_Domain)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_domain"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_language

Displays the three-letter abbreviation of the language currently being used.

---

### Example

```
Script(Speech_To_Text_Setting_Language)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_language"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_language\\_short](#), [stt\\_language\\_long](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),  
[Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_language\_short

Displays the three-letter abbreviation of the language currently being used.

---

### Example

```
Script(Speech_To_Text_Setting_Language_Short)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_language_short"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_language](#), [stt\\_languge\\_long](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_language\_long

Displays the full name of the language currently being used. For example, American English instead of ENU.

---

### Example

```
Script(Speech_To_Text_Setting_Language_Long)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_language_long"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_language](#), [stt\\_language\\_short](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_frequency

Displays the sampling frequency.

---

### Possible Values

8KHz  
11KHz  
16KHz

---

### Example

```
Script(Speech_To_Text_Setting_Frequency)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_frequency"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[tts\\_frequency](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_size

Displays the size of the speech-to-text engine being used.

---

### Possible Values

Full  
Compact  
Ultra Compact

---

### Example

```
Script(Speech_To_Text_Setting_Size)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_size"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_timeout

Indicates the total milliseconds for the system to wait before responding to the speaker.

---

### Example

```
Script(Speech_To_Text_Setting_Timeout)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_timeout"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_idle\\_timeout](#), [stt\\_silence](#), [stt\\_fx\\_silence](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),  
[Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_idle\_timeout

Indicates the total milliseconds for the engine to continue collecting results following the last result or `stt_timeout`.

---

### Possible Values

The default value is 10000 ms (10 seconds).

---

### Remarks

If any changes (settings, grammar, etc.) are made during the `stt_idle_timeout` period, the results generated during the period will be discarded.

---

### Example

```
Script(Speech_To_Text_Setting_Idle_Timeout)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_idle_timeout"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_timeout, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

## stt\_silence

Indicates the milliseconds of silence used to indicate that the user is done speaking.

---

### Example

```
Script(Speech_To_Text_Setting_Silence)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_silence"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_timeout](#), [stt\\_idle\\_timeout](#), [stt\\_fx\\_silence](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),  
[Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_fx\_silence

Indicates the milliseconds of silence used to indicate that the user is done speaking.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Silence)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_silence"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_timeout](#), [stt\\_idle\\_timeout](#), [stt\\_silence](#), [Speech\\_Get\\_Setting](#),  
[Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#),  
[Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_expanded

Use this to get the confidence value along with the speech-to-text result.

---

### Possible Values

1 (enabled)  
0 (disabled)

The default value is 0 (disabled).

---

### Return Value

If this setting is 1, speech-to-text actions return a string with each likely speech-to-text result, followed by a newline character, the confidence value for the result, and another newline character.

---

### Remarks

There may be more than one result returned; however, the first result is the one with the highest confidence value. You can use this information to determine the appropriate `stt_threshold` and `stt_confidence` values.

---

### Example

```
Script(Speech_To_Text_Setting_Expanded)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_expanded"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
```

```
strMessage = String_Combine(strDescription, "; setting  
value:")  
strMessage = String_Combine(strMessage,  
Number_To_String_Decimal(nSettingValue))  
Ask_OK(strMessage, strSetting)  
Return
```

---

## See Also

[stt\\_confidence](#), [stt\\_threshold](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_confidence

Indicates the minimum amount of difference between the confidence for the most likely and next-most likely items that will be accepted.

---

### Possible Values

The default value is 1.

---

### Remarks

If the difference is less than the set value, the result will be discarded and the speech-to-text action will report that it failed.

You may want to use different values for different grammars.

---

### Example

```
Script(Speech_To_Text_Setting_Confidence)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_confidence"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_expanded, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

## stt\_fx\_detect\_start

Indicates the action the speech engine should take before attempting to determine what the user is saying.

---

### Possible Values

1 (enabled)

0 (disabled)

The default value is 1 (enabled).

---

### Remarks

If the setting is 1, the speech engine will wait until it detects the user is speaking; if the setting is 0, the speech engine will expect that the user should start speaking immediately.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Detect_Start)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_detect_start"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
value:")
    strMessage = String_Combine(strMessage,
Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## stt\_threshold

Indicates the minimum amount of confidence for the most-likely result that will be accepted.

---

### Possible Values

The default value is 4500.

---

### Remarks

If the confidence is less than the set value, the result will be discarded and the speech-to-text action will report that it failed.

You may want to use different values for different grammars.

---

### Example

```
Script(Speech_To_Text_Setting_Threshold)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_threshold"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_expanded, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

## stt\_fx\_threshold

Indicates the amount of energy the microphone input must have before the speech detection (`stt_fx_detect_start`) is activated.

---

### Possible Values

0 (-72dB) to 9000 (18dB)

The default value is 2200 (-50dB)

---

### Remarks

Each increase of 100 is equal to 1dB.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Threshold)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_threshold"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_fx\_detect\_start, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

## stt\_save\_threshold

Directs the speech engine to save the state if the result confidence is greater than the result confidence for `stt_threshold` and `stt_save_threshold` combined.

---

### Possible Values

The default value is 1000.

---

### Example

```
Script(Speech_To_Text_Setting_Save_Threshold)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_save_threshold"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_threshold](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_fx\_min\_duration

Indicates the minimum duration (in ms) of speech before speech detection is activated.

---

### Possible Values

10ms - 400 ms

The default value is 60ms.

---

### Remarks

The speech must also have the amount of energy required by the `stt_fx_threshold` setting.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Min_Duration)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_min_duration"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_fx\_threshold, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_fx\_sensitivity

Indicates the speech detection sensitivity.

---

### Possible Values

Any number from 0 to 1000

The default value is 50.

---

### Remarks

A higher value means speech is more easily detected.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Sensitivity)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_sensitivity"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_fx\_detect\_start, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_volume

Indicates the current volume of the microphone input.

---

### Possible Values

Any number from 0 (lowest) to 100 (highest).

---

### Remarks

This setting is not supported by all mobile devices.

---

### Example

```
Script(Speech_To_Text_Setting_Volume)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_volume"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

[stt\\_calibrate](#), [stt\\_fx\\_microphone](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

## **stt\_calibrate**

Opens the microphone calibration wizard.

---

### **Example**

```
Script(Alt_F9_Microphone_Calibrate)
Activate(On_Key, 0x78, Alt)
  Comment: Pressing Alt-F9 displays the microphone
  calibration dialog.
  Speech_Change_Setting("stt_calibrate", 1)
Return
```

---

### **See Also**

[stt\\_volume](#), [stt\\_fx\\_microphone](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_grammar\_optimize

Indicates the action the speech engine should take when a grammar is loaded.

---

### Possible Values

1 (on)

0 (off)

The default value is 1 (on).

---

### Remarks

If the setting is 1, the speech engine will attempt to optimize the grammar when it is loaded. It is recommended that you enable this setting unless you have a complicated grammar that is not working correctly.

---

### Example

```
Script(Speech_To_Text_Setting_Grammar_Optimize)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_grammar_optimize"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[stt\\_context\\_optimize](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_grammar\_phonetic

Indicates whether the grammar is allowed to contain information on phonetic transcription of words.

---

### Possible Values

1 (on)

2 (off)

The default value is 1 (on).

---

### Remarks

If you do not need this feature, it is recommended that you turn it off to save memory.

---

### Example

```
Script(Speech_To_Text_Setting_Grammar_Phonetic)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_grammar_phonetic"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_grammar\_nonterminal, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_grammar\_nonterminal

Indicates whether an error will be generated if a non-terminal is found in the definition section of any modifiable rule.

---

### Possible Values

1 (on)

0 (off)

The default value is 0 (off).

---

### Remarks

If you do not need this feature, it is recommended that you turn it off to save memory.

---

### Example

```
Script(Speech_To_Text_Setting_Grammar_Nonterminal)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_grammar_nonterminal"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_grammar\_phonetic, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_context\_optimize

Indicates the action the speech engine should take when a grammar is loaded.

---

### Possible Values

1 (on)

0 (off)

The default value is 1 (on).

---

### Remarks

If the setting is 1, the speech engine will attempt to optimize the grammar context when it is loaded.

It is recommended that you enable this setting unless you have a complicated grammar that is not working correctly.

---

### Example

```
Script(Speech_To_Text_Setting_Context_Optimize)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_context_optimize"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[stt\\_grammar\\_optimize](#), [Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#),  
[Speech\\_Get\\_Setting\\_Max](#), [Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#),  
[Speech\\_From\\_Text](#), [Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_processing

Indicates the action the speech engine should take when returning a grammar result.

---

### Possible Values

1 (on)

0 (off)

The default value is 1 (on).

---

### Remarks

If the setting is 1, the speech engine will return the semantic result (if available) instead of the actual phrase spoken by the user.

This setting is useful for grammars that incorporate bracket ({} ) directives.

---

### Example

```
Script(Speech_To_Text_Setting_Processing)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_processing"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_save\_session\_delay

Indicates the total milliseconds for the speech engine to wait before saving the next current state.

---

### Possible Values

The default value is 30000 ms (30 seconds).

---

### Remarks

The speech engine will save regularly, allowing it to adapt to the speaker and revert back to a saved state if necessary.

---

### Example

```
Script(Speech_To_Text_Setting_Save_Session_Delay)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_save_session_delay"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_reset\_session\_delay, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_reset\_session\_delay

Indicates the total milliseconds for the speech engine to wait for a valid response before reverting back to the last saved state.

---

### Possible Values

The default value is 120000 ms (2 minutes).

---

### Remarks

This setting prevents the performance from degrading if the user does not speak for a long period of time.

---

### Example

```
Script(Speech_To_Text_Setting_Reset_Session_Delay)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_reset_session_delay"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
    Ask_OK(strMessage, strSetting)
Return
```

---

### See Also

stt\_save\_session\_delay, Speech\_Get\_Setting, Speech\_Get\_Setting\_Value\_Desc,  
Speech\_Get\_Setting\_Max, Speech\_Find\_Setting\_Value, Speech\_To\_Text,  
Speech\_From\_Text, Speech\_Setting\_Available, Speech\_Change\_Setting

---

## stt\_special\_sounds

Indicates how the speech engine should interpret special sounds.

---

### Possible Values

1 (on)

0 (off)

The default value is 0 (off).

---

### Remarks

If the setting is 1, the speech engine will examine sounds to determine if they are more likely to correspond to a special sound (empty pauses, coughing, etc.) than a valid grammar result.

If your grammar consists mostly of multi-syllable words or phrases, enabling this setting will result in fewer low-confidence results. However, enabling this setting may result in one- or two-syllable words (such as "yes," "two," etc.) being rejected.

---

### Example

```
Script(Speech_To_Text_Setting_Special_Sounds)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_special_sounds"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_fx\_microphone

Tells the speech engine the distance between the user and the microphone.

---

### Possible Values

0 (closest)  
1 (furthest))

The default value is 0 (closest).

---

### Remarks

The default value of 0 indicates that the user's mouth is next to the microphone. A value of 1 should be used if the speaker's mouth will be located at least several inches away from the microphone.

---

### Example

```
Script(Speech_To_Text_Setting_Fx_Microphone)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_fx_microphone"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
        Speech_Get_Setting_Value_Desc(strSetting,
        nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)

---

## stt\_priority

Determines how aggressively the microphone input is collected and speech analysis is performed.

---

### Possible Values

0 (low)  
1 (medium)  
2 (high)  
3 (critical)

The default value is 1 (medium).

---

### Remarks

You can increase the priority if the results are taking a long time to process, and decrease the priority if you experience issues such as the device locking up or network connections being dropped during speech-to-text processing.

---

### Example

```
Script(Speech_To_Text_Setting_Priority)
String(strSetting)
String(strDescription)
String(strMessage)
Number(nSettingValue)
Activate(From_Menu)
    strSetting = "stt_priority"
    nSettingValue = Speech_Get_Setting(strSetting)
    strDescription =
    Speech_Get_Setting_Value_Desc(strSetting,
    nSettingValue)
    strMessage = String_Combine(strDescription, "; setting
    value:")
    strMessage = String_Combine(strMessage,
    Number_To_String_Decimal(nSettingValue))
```

```
Ask_OK(strMessage, strSetting)  
Return
```

---

**See Also**

[Speech\\_Get\\_Setting](#), [Speech\\_Get\\_Setting\\_Value\\_Desc](#), [Speech\\_Get\\_Setting\\_Max](#),  
[Speech\\_Find\\_Setting\\_Value](#), [Speech\\_To\\_Text](#), [Speech\\_From\\_Text](#),  
[Speech\\_Setting\\_Available](#), [Speech\\_Change\\_Setting](#)