



**Wavelink Telnet Client
Scripting Reference Guide**

tn-rg-script-20080601

Revised 6/01/08

Copyright © 2008 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
Website: <http://www.wavelink.com>

Email: 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

Chapter 1: Introduction	9
Document Assumptions	9
Document Conventions	10
About Telnet Client Scripting.....	10
Overview of the Scripting Process	11
 Chapter 2: Launching the Script Editor	 13
Launching the Script Editor from Avalanche Manager	13
Launching the Script Editor from Avalanche MC.....	14
 Chapter 3: Creating Scripts	 17
Creating Scripts Manually	18
Configuration Overview	18
Naming Scripts	19
Selecting Activation Methods	19
Select from Menu	20
On Key Combination	21
When Session Connects	22
On Barcode, MSR or RFID Scan	23
On Screen Update	25
Creating Script Code	27
Creating Variables	27
Persistent Variables	29
Selecting Host Profiles	29
Performing Script Capturing	31
Creating Scripts From Text.....	33
Importing a Text File	33
Building Scripts with the Text Editor	34
Editing Scripts	35
Importing Scripts	36
Saving and Exporting Scripts.....	37
Deploying Scripts.....	38
Syncing Scripts.....	38
Creating Log Files	39
Entering the Logging_On Action	39
Entering the Logging_Off Action	40
Script Nesting.....	41
Field Data ID Feature	42
Adding a Field Data ID or Symbology	42
Editing a Field Data ID or Symbology	45
 Chapter 4: Executing Scripts	 47
Select From Menu	47

On Key Combination.	48
When Session Connects	48
On Barcode, MSR, or RFID Scan	49
On Screen Update	49
From Web Pages.	49
Launching Scripts from Web Pages Example 1	49
Launching Scripts from Web Pages Example 2	49

Chapter 5: Building Scripts Manually 51

Launching the Script Editor	52
Naming the Script and Selecting the Activation Method	52
Building the Script Code.	52
Verifying the Script Starts on the Correct Screen	52
Entering the User Name and Password	59
Verifying the Screen and Navigating Menus	63

Appendix A: Examples 71

Example 1: Beep	71
Example Code	71
Notes	71
Example 2: Escape Sequence	72
Example Code	72
Notes	72
Example 3: Request Information	72
Example Code	73
Notes	73
Example 4: Display Screen Button.	73
Example Code	73
Notes	73
Sample Voice-Enabled Emulation Scripts	74
Play_Screen Sample Script	74
Get_Number_Test Sample Script	74
Get_Number Sample Script	74
Speech_Button_Demo Sample Script	75

Appendix B: Actions 77

No Return Values.	77
Blank Line and Comment Actions	78
Blank_Line	78
Comment	78
Goto Support Actions	78
Goto	78
Label	78
Macro Exiting	78
Return	78
Abort	78

Abort_All	78
Disconnect	78
Exit_Application	78
Conditionals	79
If	79
If_Not	79
Else	79
End_If	79
While	79
While_Not	79
End_While	80
Continue	80
Break	80
General Queries	80
Ask_OK	80
Send Characters	80
Keypress_String	80
Keypress_Key	80
Scan_String	80
Set_Cursor_Position	80
Message	80
Message	80
Message_Clear	81
Sounds	81
Beep	81
Play_Sound	81
Waiting	81
Wait_For_Screen_Update	81
Delay	81
Logging	81
Logging_On	81
Logging_Off	82
Call Other Macros	82
Call	82
Screen Buttons	82
Button_Bitmap_Create_Emulation	82
Button_Bitmap_Create_View	82
Button_Create_Emulation	83
Button_Create_View	83
Button_Remove	83
Button_Remove_All	83
Reboot	83
Boolean Values	84
Boolean Assignments	84
Boolean_Set	84
Boolean_Not	84

Boolean_And	84
Boolean_Or	85
Boolean Comparisons	85
Boolean_Equal	85
Boolean_Not_Equal	85
String Comparisons	85
String_Empty	85
String_Less_Than	85
String_Less_Than_Or_Equal	85
String_Equal	85
String_Greater_Than_Or_Equal	86
String_Greater_Than	86
String_Not_Equal	86
Field Identifiers and Data	86
Set_Field_Data_ID	86
Set_Field_Symbology_ID	87
Get_Field_Symbology_Operator	87
Set_Field_Append_Scan_Data	87
Set_Field_Com_Data_Field	87
Set_Field_Prefix_Scan_Data	87
Get_Field_Append_Scan_Data	88
Integer Comparison	88
Number_Less_Than	88
Number_Less_Than_Or_Equal	88
Number_Equal	88
Number_Greater_Than_Or_Equal	88
Number_Greater_Than	88
Number_Not_Equal	88
General Queries	88
Ask_OK_Cancel	88
Ask_Yes_No	88
Suspend	88
Search the Screen	89
Search_Screen	89
WEB Emulation Commands	89
Web_Navigate	89
Web_Navigate_Frame	89
Web_Navigate_Post_Data	89
Web_Scripting	89
Web_Search_Source	90
Speech Commands	90
Speech_From_Text_Available	90
Speech_From_Text	90
Speech_To_Text_Available	90
Speech_To_Text	90
Speech_To_Text_No_Wait	90

Speech_To_Text_Cancel	90
Speech_Setting_Available	91
Speech_Change_Setting	91
Speech_Get_Setting	91
Speech_Get_Setting_Max	91
Speech_Find_Setting_Value	91
Speech_Get_Setting_Value_Desc	91
Speech_To_Text_Get_User_Name	91
Speech_To_Text_Change_User_Name	91
String Values	91
Get System Information	92
Get_MAC_Address	92
Get_IP_Address	92
Get_Field_Symbology_ID	92
Get_Screen_Text	92
Get_Screen_Text_Length	92
Get_Screen_Text_Columns	92
Get_Workstation_ID	93
Scanner Information	93
Get_Scan_Type_Name	93
ESC Sequence Support	93
Escape_Sequence	93
String Variable Assignments	93
String_Set	93
String_Combine	93
String_Left	93
String_Right	93
String_Middle	94
String_Upper	94
String_Lower	94
String_Replace	94
String_Only_Characters	94
String_Strip_Characters	94
String_Trim_Spaces_Start	94
String_Trim_Spaces_End	94
Number_To_String_Binary	95
Number_To_String_Octal	95
Number_To_String_Decimal	95
Number_To_String_Hexadecimal_Lowercase	95
Number_To_String_Hexadecimal_Uppercase	95
Ask_String	95
Ask_String_Password	95
Ask_String_Uppercase	95
Ask_String_Lowercase	95
Number to Character Conversion	96
Number_To_Character	96

Field Identifiers and Data	96
Get_Field_Data_ID	96
Get_Field_Prefix_Scan_Data	96
Integer Values	96
Get System Information	97
Get_Screen_Columns	97
Get_Screen_Rows	97
Get_Position_Column	97
Get_Position_Row	97
Get_Session_Number	97
Get_Time	97
Get_Time_Since_Reset	97
Get_Field_Index	97
Get_Num_Fields	97
Get_Field_Index_Row_Text	98
Get_Field_Index_Column_Text	98
Get_Field_Row	98
Get_Field_Column	98
Get_Field_Length	98
Get_Num_Field_Data_IDs	98
Get_Num_Field_Symbology_IDs	98
Get_Field_Com_Data_Field	99
Scanner Information	99
Get_Scan_Type_Value	99
General Queries	99
String Handling	99
String_Length	99
String_Find_First	99
String_Find_Last	99
Integer Assignments	100
Number_Set	100
Number_Plus	100
Number_Minus	100
Number_Multiply	100
Number_Divide	100
Number_Divide_Remainder	100
Convert Strings to Integers	100
String_To_Number_Binary	100
String_To_Number_Octal	100
String_To_Number_Hexadecimal	101
Ask User for Integer	101
Ask_Number	101
Number/Character Conversion	101
Character_To_Number	101

Appendix D: Wavelink Contact Information	107
Index	109

Chapter 1: Introduction

This document provides information about creating and executing scripts using Telnet Client. This chapter provides an overview of the document and scripting features. It includes:

- Document Assumptions
- Document Conventions
- About Telnet Client Scripting

Document Assumptions

This document is written with the assumption the reader and user of the Telnet Client possess:

- Knowledge of wireless networks and wireless networking protocols.
- Knowledge of TCP/IP, including IP addressing, subnet masks, routing, BootP/DHCP, WINS, and DNS.
- Knowledge of Wavelink Telnet Client.
- Knowledge or rudimentary experience with programming/scripting languages.

Document Conventions

The following table lists the document conventions used in this manual.

`Courier New`

Any time you type specific information into a text box (such as a file name), that option appears in the `Courier New` text style. This text style is also used for any keyboard commands that you might need to press.

Examples:

Type `Enter` to continue.

Press `CTRL+ALT+DELETE`.

Bold

Any time you interact with an option (such as a button or descriptions of different options in a dialog box), that option appears in the **Bold** text style.

Examples:

Click **Open** from the **File** Menu.

Select the **Update** option.

Italics

Any time this document refers to another section within the document, that section appears in the *Italics* text style. This style is also used to refer to the titles of dialog boxes.

Examples:

See *Naming Scripts* on page 19 for more information.

The *Script Editor* dialog box appears.

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 included in Telnet Client 5.1 and later versions.

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, from Avalanche Manager, or from Avalanche MC.
- 2 Create scripts using the Script Editor.** You can 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.

NOTE You can only run scripts while a Telnet session is connected to a host. If the connection drops, the script is terminated. If you switch between sessions, the script running in the first session remains suspended until that session is active again.

Chapter 2: Launching the Script Editor

The method you use to launch the Script Editor depends on where you are deploying the Telnet Client. This chapter provides information about the following tasks:

- Launching the Script Editor from Avalanche Manager
- Launching the Script Editor from Avalanche MC

Launching the Script Editor from Avalanche Manager

If you are using Avalanche Manager to deploy the Telnet Client, you launch the Script Editor from the Avalanche Manager. Scripts created by or imported into the Script Editor are automatically deployed to the remote devices.

To launch the Script Editor from Avalanche Manager:

- 1 Install the Telnet Client package in Avalanche Manager.
- 2 From the Tree View in the Avalanche Console, right-click the Telnet Client software package.
- 3 Select **Configure Package > Script Editor**.

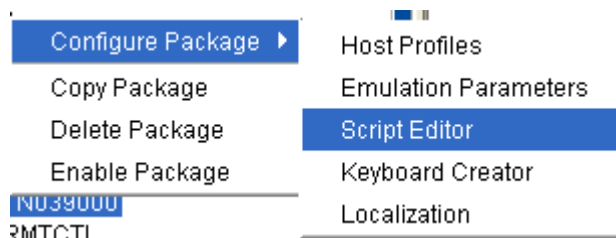


Figure 2-1. Launching the Script Editor from Avalanche Manager

The Script Editor appears.

- 4 Click **Add** to access the *Script Editor Configuration* dialog box.

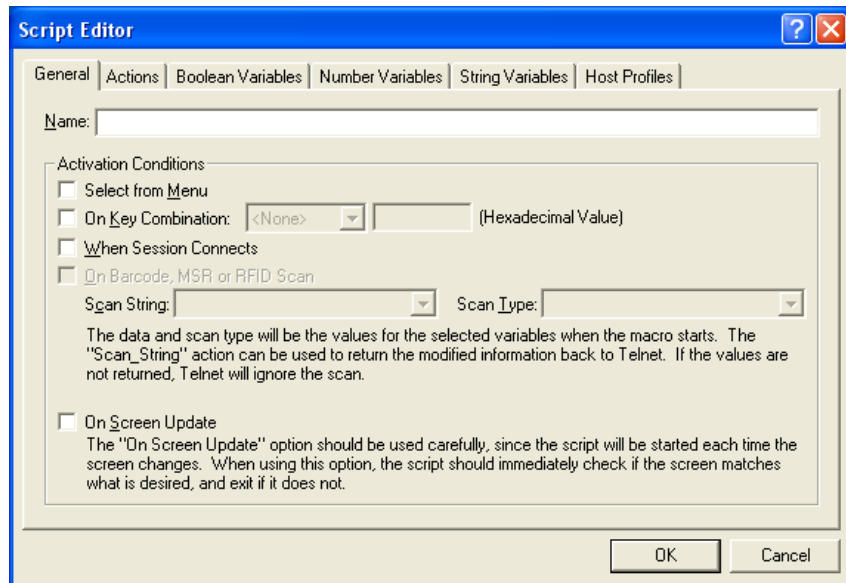


Figure 2-2. Script Editor Configuration Dialog Box

From this dialog box, you can configure and create scripts. For information about script creation, refer to *Chapter 3: Creating Scripts* on page 17.

Launching the Script Editor from Avalanche MC

If you are deploying the Telnet Client from Avalanche MC, you can launch the Script Editor from the Avalanche MC Console.

To launch the Script Editor from Avalanche MC:

- 1 Install the software package to Avalanche MC.
- 2 Select the software profile that includes the Telnet Client package.
- 3 Select the **Software Packages** tab, then click the **Enter Edit Mode** button.
- 4 Select the Telnet Client package and click **Configure**.

The *Configure Software Package* dialog box appears.

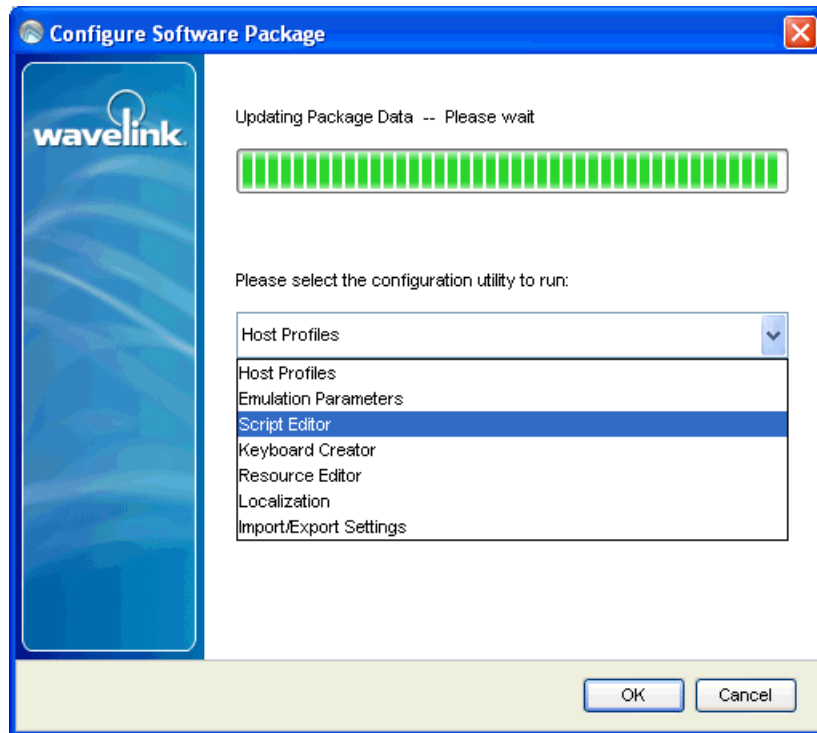


Figure 2-3. *Configuring the Software Package*

- 5 Select **Script Editor** and click **OK**.

The Script Editor appears.

- 6 Click **Add** to access the *Script Editor Configuration* dialog box.

From this dialog box you can create and configure scripts. For more information about script creation, refer to *Chapter 3: Creating Scripts* on page 17.

Chapter 3: Creating Scripts

There are three ways you can create scripts:

- **Manually.** Using this method, you build the script code from scratch using the Script Editor.
- **Script Capturing.** Using this method, you generate the script code by enabling script capturing and performing the action you want the script to automate. The Script Editor records the key presses and cursor movements and saves the information as script code. You can edit the captured code to customize the way the script runs.
- **From Text.** Using this method, you generate the script code by importing text from any text editor, or entering text in the Script Editor.

Once you create a script, you can save and deploy the script to the mobile devices you want to run the script.

This section provides information about creating scripts, including:

- Creating Scripts Manually
- Performing Script Capturing
- Creating Scripts From Text
- Editing Scripts
- Importing Scripts
- Saving and Exporting Scripts
- Deploying Scripts
- Syncing Scripts
- Creating Log Files
- Script Nesting
- Field Data ID Feature

NOTE Screen captures in this document may differ according to device type.

Creating Scripts Manually

When you create a script manually, you build the code for the script line by line in the Script Editor. This section provides information about creating scripts manually and includes the following information:

- Configuration Overview
- Naming Scripts
- Selecting Activation Methods
- Creating Script Code
- Creating Variables
- Selecting Host Profiles

Configuration Overview

The following steps provide an overview of how you create a script. The subsequent sections in this chapter describe each step in more detail.

To build a script:

- 1 Name the script.
- 2 Select an activation method.
- 3 Build the script code. In the **Actions** tab, create the code, line-by-line, that describes what actions you want the script to perform.
- 4 Create any variables that you need for your script in the **Boolean Variables**, **Number Variables**, or **String Variables** tabs.
- 5 Assign host profiles that can perform the script.

Your script is complete and ready to activate upon a Telnet Client-to-host connection.

Naming Scripts

Give scripts a unique name according to the action the script performs. The script name is the name you will select from when activating the scripts. You can name your script in the **General Settings** tab.

To name scripts:

- 1 Launch the Script Editor.
- 2 Click **Add** to access the *Script Editor Configuration* dialog box.
- 3 In the **General Settings** tab, enter the name of the script.
- 4 Click **OK** to save the name and exit the *Script Editor Configuration* dialog box.

Selecting Activation Methods

The Activation Method determines the way you execute the script from the Telnet Client. A script with no activation method selected can only be called by another script. It cannot activate alone.

Activation methods are selected in the **General Settings** tab of the *Script Editor Configuration* dialog box. This section provides information about the different activation methods including:

- Select from Menu
- On Key Combination
- When Session Connects
- On Barcode, MSR or RFID Scan
- On Screen Update

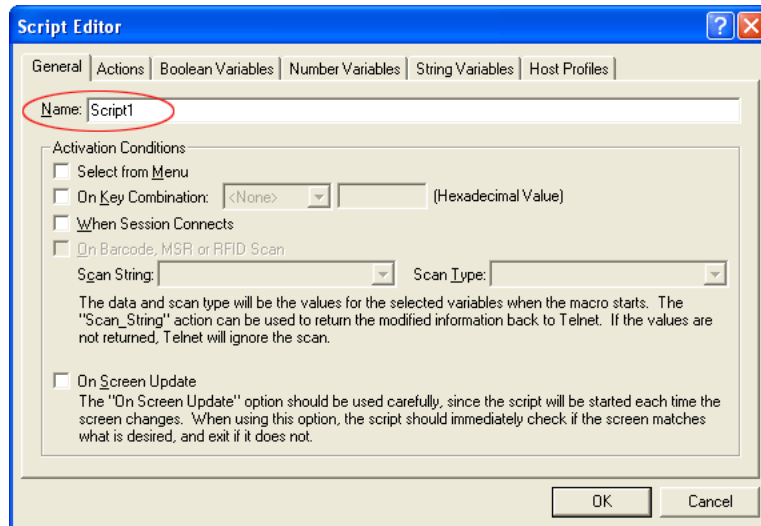


Figure 3-1. *Entering the Script Name*

Select from Menu

The **Select from Menu** option places a script execution selection in the Telnet Client menu.

To configure the **Select from Menu** method:

- 1 Select the **General** tab or the **Activate** tab in the Script Editor.
- 2 Enable the **Select from Menu** option.

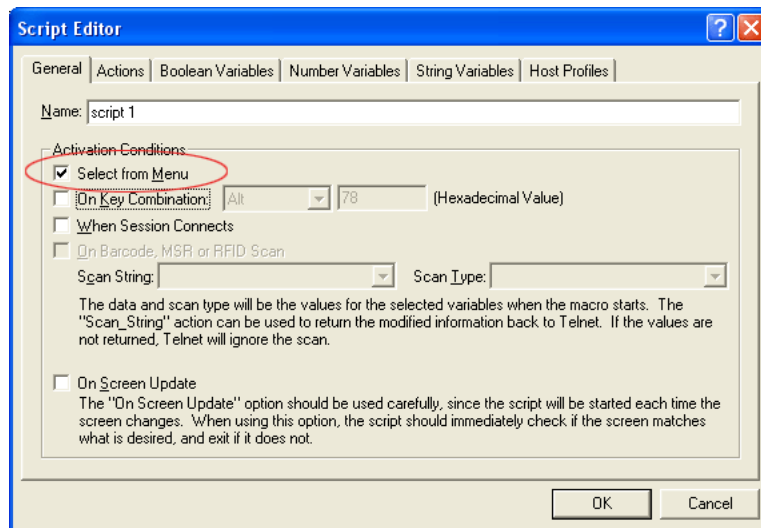


Figure 3-2. *Select from Menu*

3 Click OK.

On Key Combination

If you enable the **On Key Combination** option, you can launch the script by pressing the specified keys.

Use the Diagnostic Utility to obtain the key value. For more information about using the Diagnostic Utility refer to the *Wavelink Telnet Client User Guide*.

To configure the On Key Combination method:

- 1** Select the **General** tab or **Activate** tab in the Script Editor.
- 2** Enable the **On Key Combination** option.

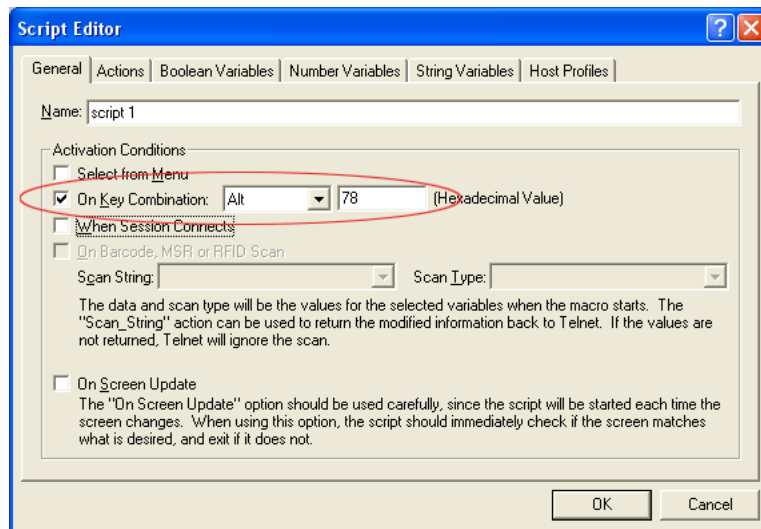


Figure 3-3. *On Key Combination*

- 3 Use the drop-down menu and text box to assign a key combination to the script.
- 4 Click OK.

When Session Connects

If you enable the **When Sessions Connects** option, the script activates when the host profile it supports is activated.

If you use this option, Wavelink strongly recommends that you limit the script to the appropriate host profiles. Because the script activates before any information appears on the emulation screen, you need to have your script wait for the appropriate screen to appear before activates. You should not have more than one script set to start when a session connects because the first script that starts will prevent any other scripts from running while it waits for the initial screen. Refer to *Selecting Host Profiles* on page 29 for more information.

To configure the When Session Connects method:

- 1 Select the **General** tab or **Activate** tab in the Script Editor.
- 2 Enable the **When Session Connects** option.

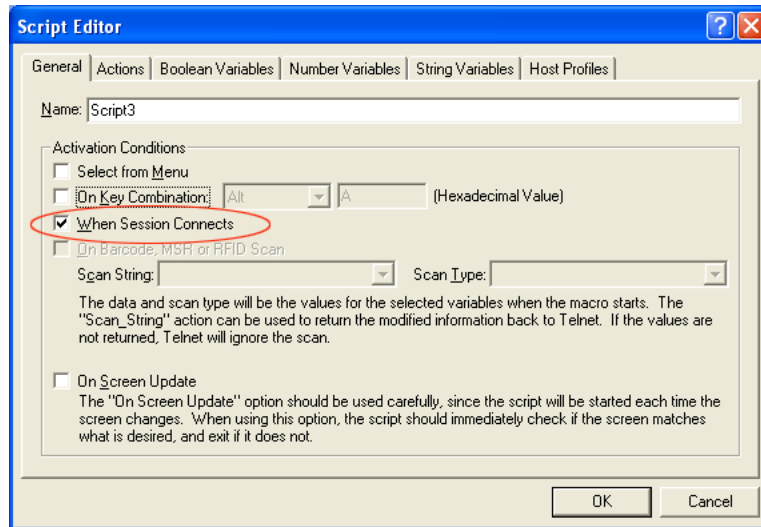


Figure 3-4. *When Session Connects*

3 Click OK.

On Barcode, MSR or RFID Scan

If you want to perform special processing on items scanned into the computer, the Scan Handler is often powerful enough to make the changes you need. The Scan Handler settings, found in the Configuration Manager, are located in **Emulation Parameters > Scanner > Common > Scan Handler**. However if the Scan Handler is insufficient, you can use a script to perform any processing you need.

Before you can activate the script for a scan, you must create a string variable and a number variable.

- The string variable allows you to obtain the initial scan data.
- The number variable allows you to obtain the type of scan data.

Refer to *Appendix C: Symbolologies and Values* on page 103 for the values of different symbolologies. You can also use the `Get_Scan_Type_Name` and `Get_Scan_Type_Value` commands to display or handle scan types. Using the `Get_Scan_Type_Value` means all types are specified in the editor and you can select the type you want to use.

Calling the `Scan_String` command before your script exits allows Telnet to handle the scanning data. Because you are specifying the data and type returned, the script can change either one. If the script exits without calling `Scan_String`, the scanned data disappears.

To configure the On Barcode, MSR, or RFID Scan method:

- 1 Create the `Scan_String` and `Scan_Type` variables.

Once you create these variables, the **On Barcode, MSR or RFID Scan** option becomes available.

Create these variables in the String Variables and Number Variables tabs. Refer to *Creating Variables* on page 27 for information on creating variables.

- 1 Select the **General** tab or **Activate** tab in the Script Editor.
- 2 Enable the **On Barcode, MSR or RFID Scan** option.
- 3 From the drop-down menu, select the **Scan_String**.
- 4 From the drop-down menu select the **Scan_Type**.
- 5 Click **OK**.

On Barcode, MSR or RFID Scan Example 1

The following is a sample script you can use if you want to insert a string (which could be just one character long) after the first six characters of any barcode at least six characters long.

A few notes about the sample script:

- **ScanData** is a string variable with the original barcode.
- **NewString** is a variable where you store the new barcode.
- **ScanType** is the number variable that keeps the type of scan data received.
- **OldLength** is an integer variable.
- **XXYY** is the string you insert.

```
OldLength=String_Length(ScanData)
If (Number_Greater_Than_Or_Equal(OldLength,6))
NewString=String_Combine(String_Left(ScanData,6), "XXYY")
```

```

)
  NewString =
String_Combine(NewString,String_Right(ScanData,
Number_Minus(OldLength,6)))
  Else
    NewString = ScanData
  End_If
  Scan_String(NewString,ScanType)
Return

```

On Barcode, MSR or RFID Scan Example 2

This example converts any DataMatrix scan values to PDF417 scan values. The **ScanData** and **ScanType** variables described for the previous example are used again.

```

If
(Number_Equal(ScanType,Get_Scan_Type_Value("DATAMATRIX")))
)
  Scan_String(ScanData,Get_Scan_Type_Value("PDF417"))
Else
  Scan_String(ScanData,ScanType)
End_If
Return

```

On Screen Update

This option activates the script (if activation is allowed) every time the text on the emulation screen changes. This includes updates from the Telnet host or when the user presses a key and the key value is shown on the screen. It is recommended that you limit the host profiles that the script supports.

The following example generates a script that enters a command each time a particular string appears on the screen:

```

Label:Start:
If
(String_Equal(Get_Screen_Text_Columns(1,1,5),"Ready",
0,FALSE))
  Keypress_String("Proceed")
  Keypress_Key("Enter")
End_If
Wait_For_Screen_Update
Goto: Start
Return

```

If the script is set to activate when the session first connects, it will work as desired with one limitation. Because it is always activated, no other scripts can be activated during the emulation session.

Here is an alternate implementation:

```
If (String_Equal(Get_Screen_Text_Columns(1,1,5),
"Ready", 0, FALSE))
  Keypress_String("Proceed")
  Keypress_Key("Enter")
End_If
Return
```

If the script is set to run each time the screen updates, you get the desired behavior. Because the script is not active all the time, other scripts can be activated as well.

NOTE Use this option carefully as it can cause a script to be executed frequently.

To configure the On Screen Update method:

- 1 Select the **General** tab or **Activate** tab in the Script Editor
- 2 Enable the **On Screen Update** option.

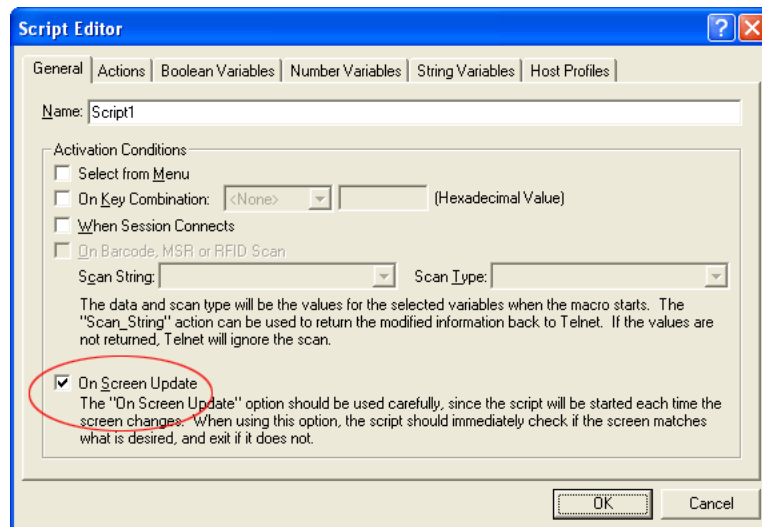


Figure 3-5. Selecting the On Screen Update Method

- 3 Click **OK**.

Creating Script Code

Once you name a script and select an activation method, you can use the **Actions** tab in the Script Editor to build the script. Refer to *Chapter 5: Building Scripts Manually* on page 51 for a detailed example of creating script code manually.

Creating Variables

There are three types of values recognized by scripting:

- Booleans (TRUE or FALSE values only)
- Numbers (integers)
- Strings

Every argument for every action is one of these three value types. Every action that returns a value returns one of these types. Variables provide a way to save the result of an action for later use as an argument for another command.

You can create and edit variables located in the corresponding tabs while editing a script. You can also create new variables while editing an action.

When a script starts, all the variables will have known values: boolean variables are FALSE, number variables are 0, and string variables are empty. One possible exception to this is when a script activates another script. Refer to *Script Nesting* on page 41 for more information.

To create variables:

- 1 Determine which type of variable you want to create: boolean, number, or string.
- 2 From the Script Editor, select the tab that corresponds with the type of variable you want to create.
- 3 Click **Add**.
- 4 In the *Edit Variable* dialog box, enter the name of the new variable.

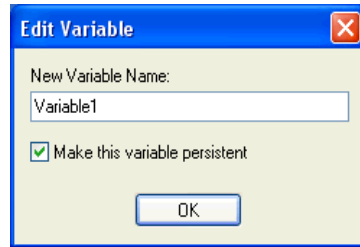


Figure 3-6. Adding a New Variable

- 5 If you would like the variable to be a persistent variable, enable the **Make this variable persistent** checkbox. For more information, refer to *Persistent Variables* on page 29.
- 6 Click **OK**.

The new variable appears in the corresponding tab.

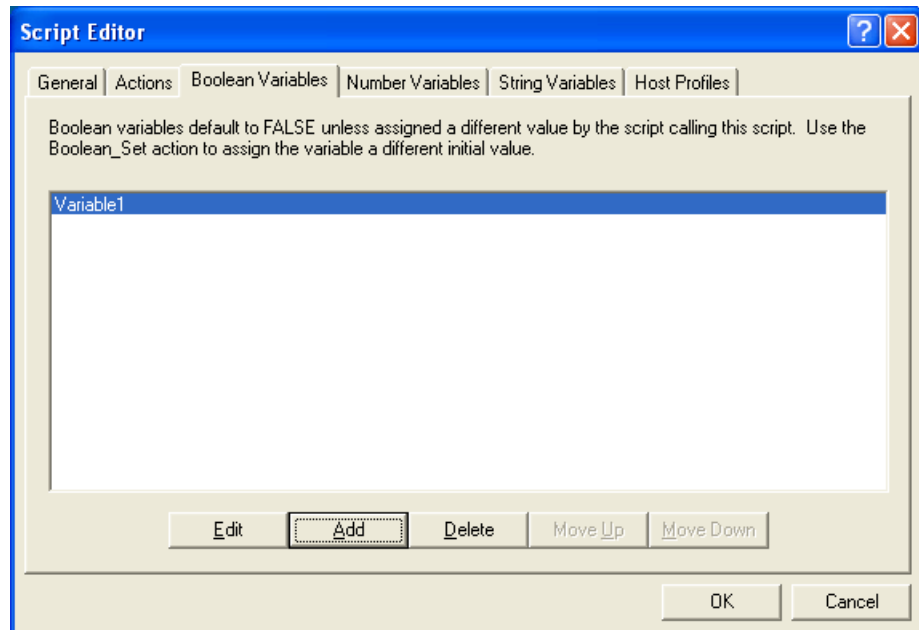


Figure 3-7. New Variable

Persistent Variables

When a variable is persistent, the value remains after the script exits. Persistent variables are not script or session specific; once a value is assigned, any script in any Telnet session can use a persistent variable to access that value.

Two scripts are considered to be referencing the same persistent variable if both scripts contain persistent variables of the same type and same name.

Because writing new values to persistent variables will slow your application, they should only be used when necessary. If you want to use a persistent variable that will change values frequently, write your script with a regular variable that only changes the value of the persistent variable before the script pauses or exits.

Selecting Host Profiles

For each script, you can specify which host profiles will be supported by that script. You may select host profiles from the **Host Profiles** tab.

If the script is generated by script capturing, limit the script to the host profile that was in use when the script was captured. The default - no host profile - allows the script to be run when any host profile is used.

To select host profiles:

- 1 From the Script Editor, select the **Host Profiles** tab.

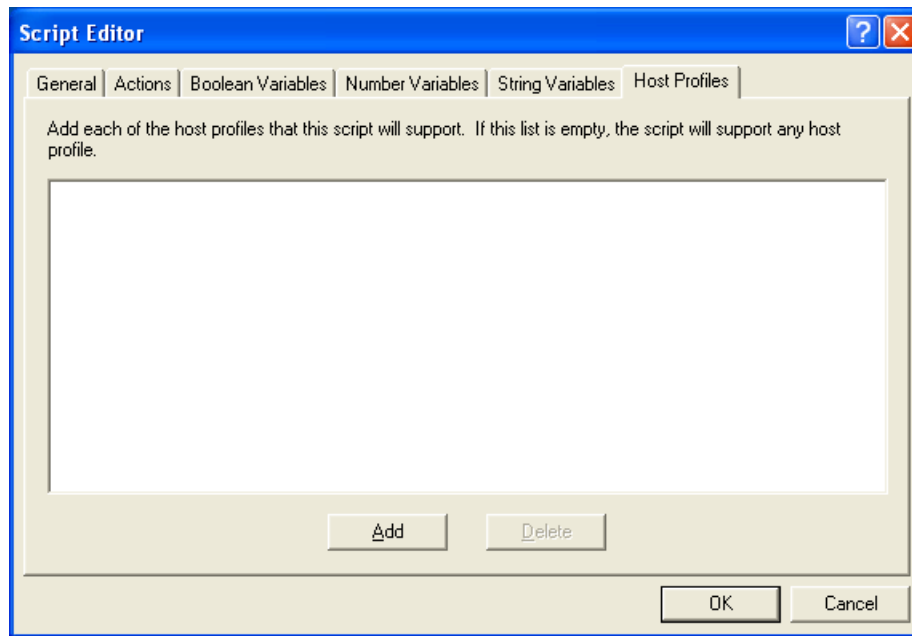


Figure 3-8. *Host Profiles Tab*

2 Click **Add**.

The *Select Host* dialog box opens.

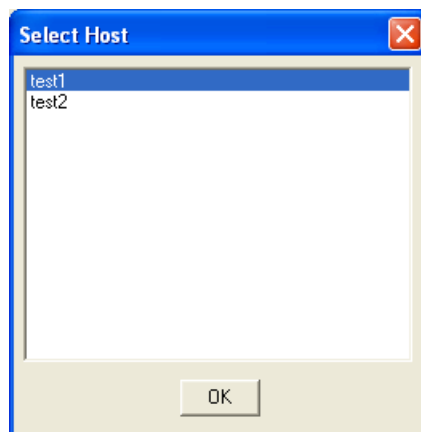


Figure 3-9. *Selecting Host Profiles*

- 3 Select which host you want to use from the list of Avalanche hosts.

NOTE If you have not created any host profiles, this dialog box will be empty.

- 4 Click **OK**.

The host appears in the **Host Profiles** tab.

Performing Script Capturing

Script capturing is an easy way to generate a script that automates actions or processes. When script capturing is activated, it captures key presses and mouse/pen cursor movements so those actions can be replayed when the script is activated.

To perform a script capture:

- 1 Position your mouse or cursor on the emulation screen you want to be at when the automated process starts.
- 2 From the **Term** or **Options** menu, select **Scripting > Start Capture**.

The *Script Capturing Initialization* dialog box appears, asking if you want to verify the current screen text.

- 3 Select **Yes** to verify the current screen text.

Select **No** if you do not want to verify the current screen text.

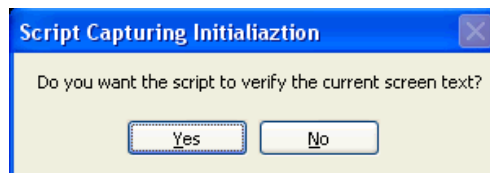


Figure 3-10. *Script Capturing Initialization Dialog Box*

Selecting **Yes** makes the captured script start with an `If_not` command that tells the script to exit if the correct screen is not currently shown. Unless you know that your script will only run from the correct screen (for example, a script that is run only when a session first starts, or a script called by another script), you should select **Yes**.

NOTE If you select **No**, click **Verify Screen Contents** and **Save Cursor Position** when you start your script capture. This causes your script to wait for Telnet to finish updating the screen before processing script actions.

- 4 Perform any actions you want to include in the script.
- 5 Each time the screen changes, click **Verify Screen Contents** button.

NOTE Some devices may only display buttons labeled **Screen**, **Cursor** and **Stop**. The **Screen** button refers to the **Verify Screen Contents** button. The **Cursor** button refers to the **Save Cursor Position** button. The **Stop** button refers to the **Stop Capturing** button.

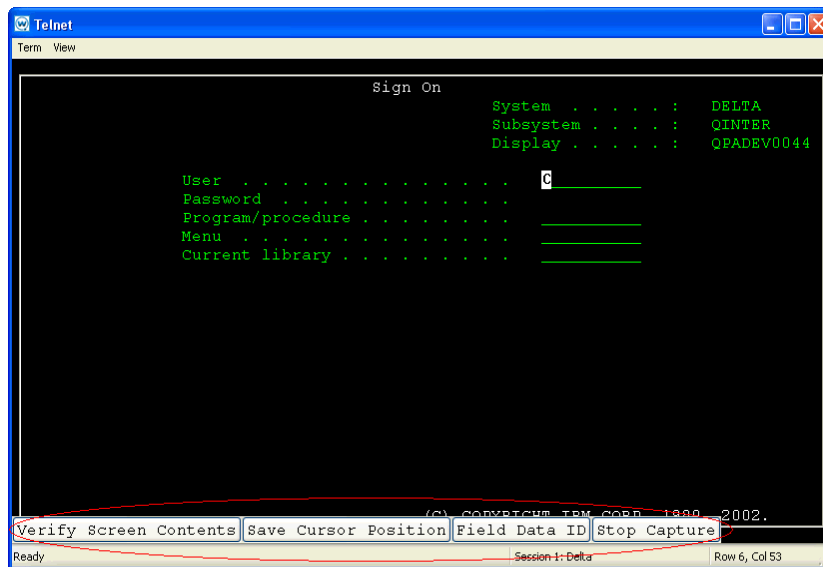


Figure 3-11. *Verify Screen Contents and Save Cursor Position Buttons*

NOTE Clicking the **Verify Screen Contents** button causes the generated script to pause and wait for the screen update. The pauses are necessary because scripts can run much faster than the interaction with the Telnet host.

- 6 When you finish capturing the behaviors you want in the script, click **Stop Capture**.

Once you capture a script, the Script Editor opens, allowing you to name the script and select an activation method. You can use the **Actions** tab to add actions for any error condition that the user may encounter.

Creating Scripts From Text

You can use the Script Editor to generate scripts from standard text. The text can be entered in the Script Editor's built-in text editor, or imported from a text-editing program such as Notepad. This section provides information about creating scripts from text, including the following:

- Importing a Text File
- Building Scripts with the Text Editor

Importing a Text File

Create a script in any standard text editor and then import the script into the Text Editor. When creating the script text, you must use the same actions and commands provided in the Script Editor.

NOTE For more information, refer to *Appendix B: Actions* on page 77.

To import a text file:

- 1 Launch the Script Editor.
- 2 Click **Import Text**.

The *Select the Script Text File* dialog box appears.

- 3 Navigate to and select the text file.
- 4 Click **Open**.

The file is imported into the Script Editor.

Building Scripts with the Text Editor

You can also create a text-based script using the Text Editor. The Text Editor will validate the script syntax when you build the script.

To build a script:

- 1 Launch the Script Editor.
- 2 Click **Add As Text**.

The Text Editor opens.

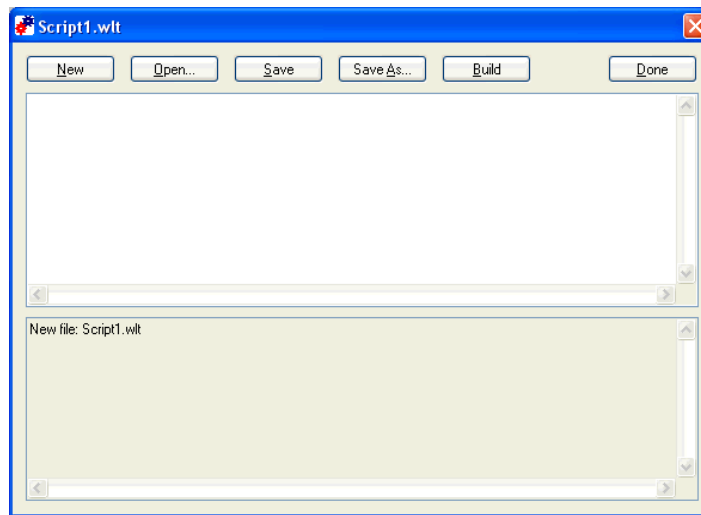


Figure 3-12. Text Editor

- 3 Enter the text into the Text Editor. You must include a script name, along with script variables and activation methods. If desired, you may also assign a Host Profile to be supported by the script.

NOTE For more information, refer to *Naming Scripts* on page 19, *Selecting Activation Methods* on page 19, *Creating Variables* on page 27, and *Selecting Host Profiles* on page 29.

- 4 After entering your text, click **Build** to verify the script text. The Text Editor displays build information in the bottom section of the window. If

the build information includes an error message, double-click the message to jump to the line that contains the error.

- 5 When you have finished building the script, click **Save** or **Save As** to save a copy of the script text.
- 6 Click **Done** to add the script to the Script Editor, or click **New** to begin building a new script.

Editing Scripts

You can edit scripts that are created manually, scripts that are generated using script capturing, and scripts created from text.

To edit scripts:

- 1 Launch the Script Editor.
- 2 Select the script you want to edit from the script list.
- 3 Click **Edit**.
- 4 Make the desired changes in the Script Editor configuration dialog box.
- 5 Click **OK** to save your changes.

Once you have completed editing the script you have two options:

- Export the script to a specified location. Refer to *Saving and Exporting Scripts* on page 37 for more information.
- Execute the script by launching the Telnet Client and performing the activation method you assigned to the script. Refer to *Chapter 4: Executing Scripts* on page 47 for more information.

To edit scripts with the Text Editor:

- 1 Launch the Script Editor.
- 2 Select the script you want to edit from the script list and click **Edit As Text**.

The Text Editor appears.

- 3 Edit the script as desired, then click **Build** to verify the script syntax.

- 4 When you have finished editing the script, click **Save** or **Save As** to save a copy of the script text.
- 5 Click **Done** to return to the Script Editor, or click **Open** to edit another script.

Importing Scripts

You can use the import button in the Script Editor to import previously created scripts.

NOTE You can only import scripts that have been created using the Script Editor.

To import a script:

- 1 From the Script Editor, click **Import**.

The *Select the Script File* dialog box opens.

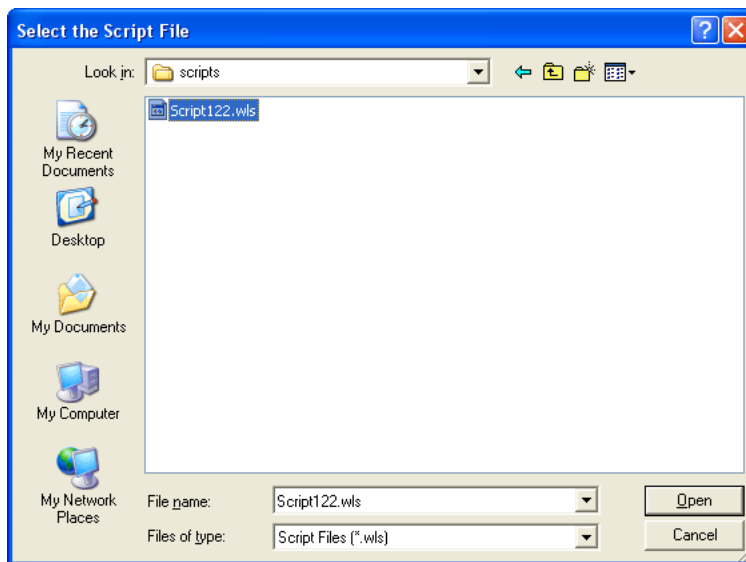


Figure 3-13. *Importing a Script File*

- 2 Navigate to and select the script file.

3 Click **Open**.

The name of the file is imported into the Script Editor.

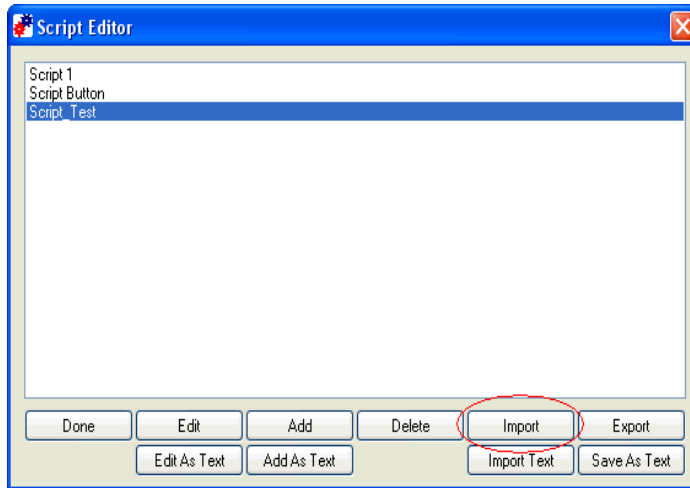


Figure 3-14. *Imported Script File*

Once you have imported the file, you can edit the script. Refer to *Editing Scripts* on page 35 for more information.

Saving and Exporting Scripts

After you finish building a script, your script is automatically saved in the Script Editor. You can also export a script and save it in a specific location on the network.

Scripts are saved as `.wls` files. Scripts can not be viewed outside the Script Editor and must be imported back into the Script Editor to view or edit.

To export a script:

- 1 From the script list, select the script you want to export.
- 2 Click **Export**.

-Or-

To save the script as text, select **Save As Text**.

The *Create the Script File* dialog box opens.

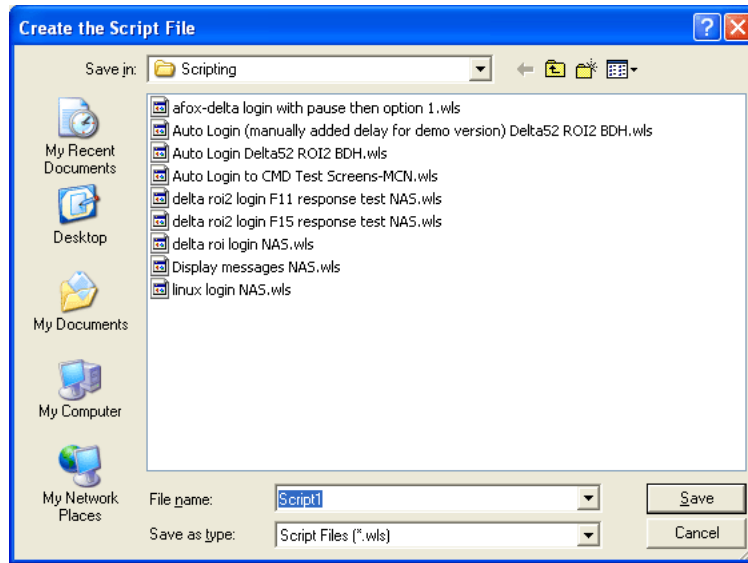


Figure 3-15. *Exporting a Script*

- 3 Navigate to the location where you want to export your script.
- 4 Click **Save**.

To view an exported script, you need to import the script into the Script Editor. Refer to *Importing Scripts* on page 36 and *Importing a Text File* on page 33 for more information.

Deploying Scripts

Scripts are deployed to the Telnet Client the next time the client syncs with the Avalanche Manager or Avalanche MC.

Syncing Scripts

The section provides information about syncing scripts that are edited or created on the mobile device to the Script Editor in Avalanche Manager or Avalanche MC.

When you create or edit a script on a mobile device, you need to sync the script to the Script Editor in Avalanche Manager. This allows you to edit and modify the scripts from the Avalanche Manager.

To sync scripts:

- 1 From the device list in Avalanche Manager, right-click the device to which you want to sync.
- 2 Select **Launch Session Monitor**.

As you connect to Session Monitor, an *Authorizing* dialog box appears and then the *Sync Script* dialog box appears.

- 3 Click **Yes** to sync your scripts from the mobile device to Avalanche Manager.

The Script Editor opens and display all scripts.

If you edited a script on the mobile device that is also saved in the Script Editor in Avalanche Manager, both scripts display in the Script Editor after syncing. The original script retains the original name. The script that was edited on the mobile device appears as `[Original Name] Telnet`.

Creating Log Files

Use the `Logging_On` and `Logging_Off` commands in your code to generate log files. Each action executed, with the values of its arguments and the results of the action, is written to the log file. When you configure the `Logging_On` action, you can set the File Path name to where the log file is stored.

When a script calls another script, the logging for the calling script is suspended. Logging resumes when the called script exits and the suspended script resumes. It is possible to have a script called by another script (or a script calling itself recursively) use the same logging file.

Entering the Logging_On Action

Include a `Logging_On` action in your script code to generate a log file. Place the `Logging_On` action at the point you want to begin logging.

To enter the Logging_On action:

- 1 From the **Actions** tab, click **Insert**.

- 2 From the **Action** drop-down menu, select **Logging_On**.
- 3 Click the **File Path** tab.
- 4 In the **Constant String** text box, enter the location where you want to store the log file.
- 5 Click the **Override Previous** tab.
- 6 Set the **Override Previous** option.

When you set the **Override Previous** option to FALSE, the latest log file will not replace the existing file. Instead, a separate log file is created for each log.

When you set the **Override Previous** option to TRUE, the most recent log file will replace the existing log file.

NOTE Because logging slows the performance of Telnet, and takes up space on your devices, you may not want to include it in end-user scripts. Set the **Override Previous** value to TRUE to keep the log files from getting too large.

- 7 Click **OK**.

The code is added to the **Actions** tab.

Entering the Logging_Off Action

If the script exits, logging automatically terminates, so you usually will not need the `Logging_Off` action. However, if you only want to log a portion of the script, you can enter a `Logging_Off` action to stop the logging.

To enter a Logging_Off action:

- 1 From the **Actions** tab, click the **Insert** button.
- 2 From the **Action** drop-down menu, select **Logging_Off**.
- 3 Click **OK**.

The code is added to the **Actions** tab.

Script Nesting

You can program a script to call other scripts or to call itself. This makes it easier to take a block of functionality and use it multiple times or to solve problems that can be described recursively.

A factorial is the product of all positive integers from 1 to the given number. For example, the factorial of 5 (usually written as 5!) = $1 \times 2 \times 3 \times 4 \times 5 = 120$. Here is an example of a script that uses recursion (a script calling itself) to calculate factorials:

```

    If (Number_Equal (ArgumentValue,1))
Comment: The factorial of 1 is 1
    Return
    End If
    If (Number_Not_Equal (ArgumentValue,0))
Comment: The factorial of X is X multiplied by the
factorial of X - 1
    Temp=ArgumentValue
    ArgumentValue=Number_Minus (Temp,1)
    Call: Factorial
    ArgumentValue <-> ArgumentValue
    ArgumentValue = Number_Multiply (Temp,ArgumentValue)
    Return
    End If
    ArgumentValue = Ask_Number ("Enter a number:",
"Factorial Calculator",1,12,0)
    Call: Factorial
    ArgumentValue <-> ArgumentValue
    Ask_OK (String_Combine ("The factorial is",
    Number_To_String_Decimal (ArgumentValue)), "Result")
    Return

```

This script uses two integer variables, `ArgumentValue` and `Temp`.

When a script calls another script, the calling script can assign values to the called script variables. The factorial example script knows it is being called recursively because the `ArgumentValue` variable is not 0. If `ArgumentValue` is 0, the script will ask for the number to calculate the factorial with. Each time the script is called, the `ArgumentValue` variable of the calling script is assigned the final value in the called script's `ArgumentValue` variable. This keeps the results of the called script's actions from being lost. (If the value were not returned, then there would be a "<--" instead of a "<->" in the Call action's argument list.) When you add the action for calling a script, you need to specify which variables in the called script will be assigned a value, and what that initial value will be.

NOTE If you wanted to be more efficient, you could create a `While` loop that performs the multiplications to calculate the factorial. You could also return the proper response for each factorial, since numbers higher than 12 exceed the maximum number value. However, there are some problems that are easiest to solve using recursion. The example above should give you an idea how you could go about using it.

Field Data ID Feature

The Field Data ID feature allows you to use scripting to configure Field Data Identifiers. Field Data Identifiers assign a unique identification, such as a letter, to each field on the screen. Any time a barcode beginning with that identifier is scanned, the information automatically populates in the corresponding field.

NOTE This feature is only available for IBM 5250 emulation.

This section provides the following information:

- Adding a Field Data ID or Symbology
- Editing a Field Data ID or Symbology

Adding a Field Data ID or Symbology

Use the *Field Data ID* dialog box to add a Data ID or Symbology to a field.

To add a Field Data ID or Symbology:

- 1 Access the IBM 5250 emulation screen.
- 2 From the **Term** menu, select **Scripting > Start Capture**.

The *Script Capturing Initialization* dialog box appears.

- 3 Click **Yes**.

The *Select Screen Text* dialog box appears.

- 4 Select the desired text string(s) from the list and click **OK**.

- 5 Place the cursor on the field where you want to add a Data ID.
- 6 From the bottom of the emulation screen, click the **Field Data ID** button.

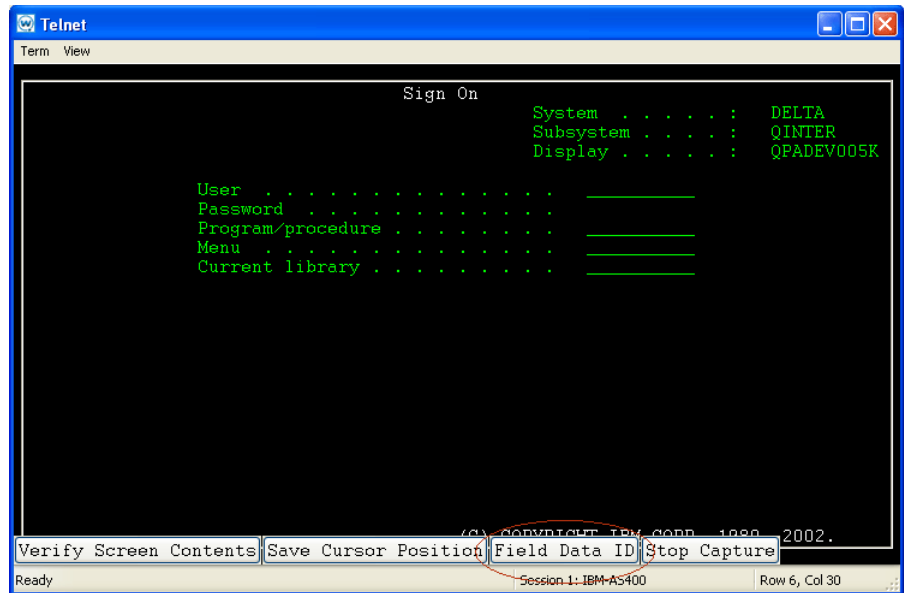


Figure 3-16. *Field Data ID Button*

The *Field Data ID* dialog box appears.

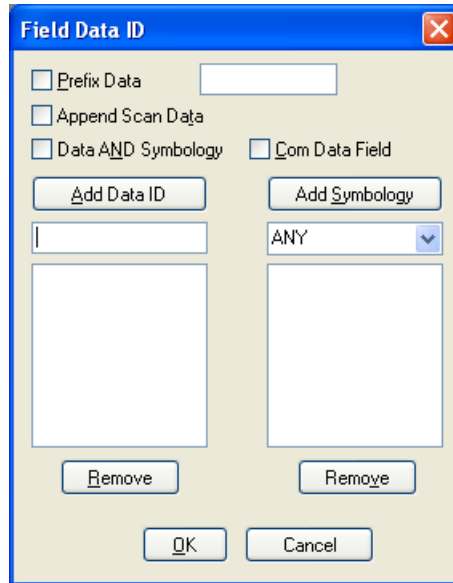


Figure 3-17. *Field Data ID Dialog Box*

7 In the text box below the **Add Data ID** button, enter the desired Data ID.

8 Click **Add Data ID**.

The new Data ID appears in the list below the text box.

9 If you want to remove a Data ID, select the desired ID from the list and click **Remove**.

10 From the drop-down menu below the **Add Symbology** button, select the desired Symbology.

11 Click **Add Symbology**.

The new Symbology appears in the list below the text box.

12 If you want to remove a Symbology, select the desired Symbology from the list and click **Remove**.

13 If you want to add a prefix to the data, enable the **Prefix Data** checkbox and enter the prefix in the available text box.

- If you want to append scan data in the field, enable the **Append Scan Data** checkbox.
- If you want to add a Data ID and a Symbology, enable the **Data AND Symbology** checkbox.
- If you want the field to be the Com Data Field for the screen, enable the **Com Data Field** checkbox.

14 Click **OK** to close the *Field Data ID* dialog box and save your changes.

Editing a Field Data ID or Symbology

If you want to make changes to a Data ID or Symbology, you must manually edit the script you created using the Field Data ID feature. For more information about editing scripts, refer to *Editing Scripts* on page 35.

Chapter 4: Executing Scripts

This section provides information about activating scripts using each of the available activation methods:

- Select From Menu
- On Key Combination
- When Session Connects
- On Barcode, MSR, or RFID Scan
- On Screen Update
- From Web Pages

For information on assigning an activation method to a script, refer to *Selecting Activation Methods* on page 19.

NOTE Screen captures may differ according to device type.

Select From Menu

If you assigned this activation method, you can activate the script from the **Term** menu of the Telnet Client.

To activate:

- 1 Launch the Telnet Client on the mobile device.
- 2 From the **Term** menu, select **Scripting > Execute Script**.

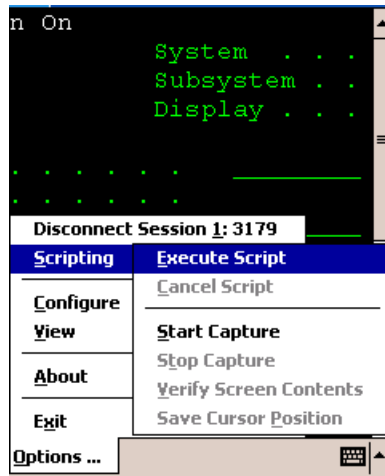


Figure 4-1. *Executing Scripts from the Menu*

- 3 If more than one script is available for the current host profile, select which script you want to use from the list.

NOTE This option is not available if a script is running for the current session or if the session is not connected.

On Key Combination

If you assigned this activation method, the script activates when you press the specified key combination (as long as it is currently possible for script to run).

To activate:

- 1 Launch the Telnet Client.
- 2 Enter the key combination you assigned to execute the script.

When Session Connects

If you assigned this activation method, the script activates when the host profile it supports is activated.

To execute when the session connects:

- 1 Launch the Telnet Client.
- 2 From the **Term** or **Options** menu, select **Connect**.
- 3 Select the host to which you want to connect.
- 4 Click **OK**.

The script runs upon connection.

On Barcode, MSR, or RFID Scan

If you assigned this activation method, the script activates with each barcode, MSR, or RFID scan.

On Screen Update

If you assigned this activation method, the script activates (if activation is allowed) every time the text on the emulation screen changes. This includes updates from the Telnet host or when the user presses a key and the key value appears on the screen.

From Web Pages

You can launch Wavelink scripts from web pages using the `wls` type, followed by the script name.

Launching Scripts from Web Pages Example 1

This example launches a script called `WebAuto` which launches when the web page first loads.

```
<title>TE70 Test1 - Launch Telnet Scripts</title>  
<meta http-equiv="OnStartup" content="wls:WebAuto">
```

Launching Scripts from Web Pages Example 2

This example launches a script called `WebClick` when a user clicks the hyperlink “here” on the web page.

```
<p>  
Click <a href="wls:WebClick">here</a> to launch the  
&quot;WebClick&quot; script.  
</p>
```

Chapter 5: Building Scripts Manually

This chapter provides information about building a script manually. The purpose of this example is to demonstrate (step-by-step) how to create a script. Values, names and variables are used for example purposes only. The example values may differ according to device type and operating system.

The following script is the example used:

```
Comment:Verify that this is the desired screen
  If_Not(String_Equal(Get_Screen_Text_Length(1, 36, 7),
"Sign On", 0, FALSE))
    Return
  End If
  Set_Cursor_Position(6, 53)

  Message("Starting Script" 3)

  Keypress_String("User Name")
  Keypress_Key("Down Arrow")
  Keypress_String("Password")
  Keypress_Key("Enter")

Comment:Wait for the desired screen.

  While_Not(String_Equal(Get_Screen_Text_Length(11,1,
16), "9. FUNCTION KEYS",0,FALSE))
    Wait_For_Screen_Update
  End_While

  Keypress_String("9")

  Message("String Done",3)
  Return
```

NOTE Screen captures may differ according to device type.

The following sections contain the tasks required to create the example script:

- Launching the Script Editor
- Naming the Script and Selecting the Activation Method
- Building the Script Code.

Launching the Script Editor

Launch the Script Editor from the Avalanche Manager or from the Telnet Client. For detailed instructions about launching the Script Editor, refer to *Chapter 2: Launching the Script Editor* on page 13.

Naming the Script and Selecting the Activation Method

In the **General** tab, name the script and select which method you want to use to activate the script. For detailed instructions, refer to *Naming Scripts* on page 19 and *Selecting Activation Methods* on page 19.

Building the Script Code

Once you name the script and select an activation method, you can begin entering the code. You build the code in the **Actions** tab. The example script performs the following actions upon completion and activation:

- Verifies that the script is starting on the right screen and sets the correct cursor position.
 - If the script is not on the correct screen, the script ends.
 - If the emulation screen is correct, the script enters a user name and password in the correct locations.
- Verifies that the emulation screen is correct.
- Selects a specific menu and exits.

The steps to enter this example code are divided into three main sections:

- 1 Verifying the Script Starts on the Correct Screen
- 2 Entering the User Name and Password
- 3 Verifying the Screen and Navigating Menus

Verifying the Script Starts on the Correct Screen

This portion of script verifies that the script is starting on the right screen and sets the correct cursor position. If the script is not on the right screen, the

script ends. Once the script verifies that it is starting on the correct screen, a message displays “Starting Script.”

To build the script code:

- 1 From the **Actions** tab, click the **Insert** button.

The *Action Editor* dialog box opens.

- 2 From the **Action** drop-down menu, select **Comment**.
- 3 Click the **Comment** tab and enter `Verify that this is the desired screen` in the **Constant String** text box.

NOTE Comments can be any string you want. This is just an example.

- 4 Click **OK**.

The code is added to the **Actions** tab.

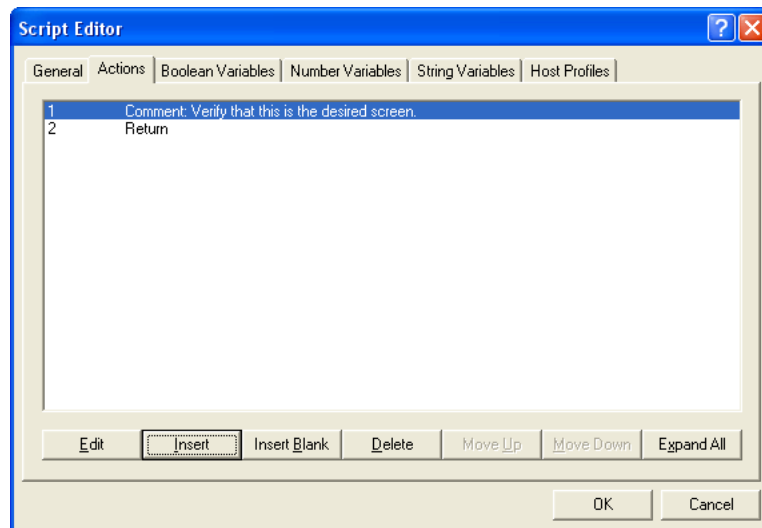


Figure 5-1. Entering a Comment

- 5 Click the **Insert** button.
- 6 From the **Action** drop-down menu, select **If_Not**.

- 7 Click the **Test** tab.
- 8 Enable the **Action** drop-down option and select **String_Equal** from the drop down menu.
- 9 Click the **Edit Action Value** button.
- 10 Click the **Test 1** tab.
- 11 Enable the **Action** drop-down option and select **Get_Screen_Test_Length** from the drop down menu.
- 12 Click the **Edit Action Value** button.
- 13 Click the **Row** tab and enter the number 1.
- 14 Click the **Column** tab and enter the number 38.
- 15 Click the **Maximum Length** tab and enter the number 7.
- 16 Click **OK**.
- 17 Click the **Test 2** tab and enter `Sign On` in the **Constant String** text box.
- 18 Click the **Maximum Length** tab and enter the number 0 in the **Constant Number** text box.
- 19 Click the **Ignore Case** tab and enable the **False** option.
- 20 Click **OK** until you return to the **Actions** tab in the Script Editor.

The code appears in the **Actions** tab.

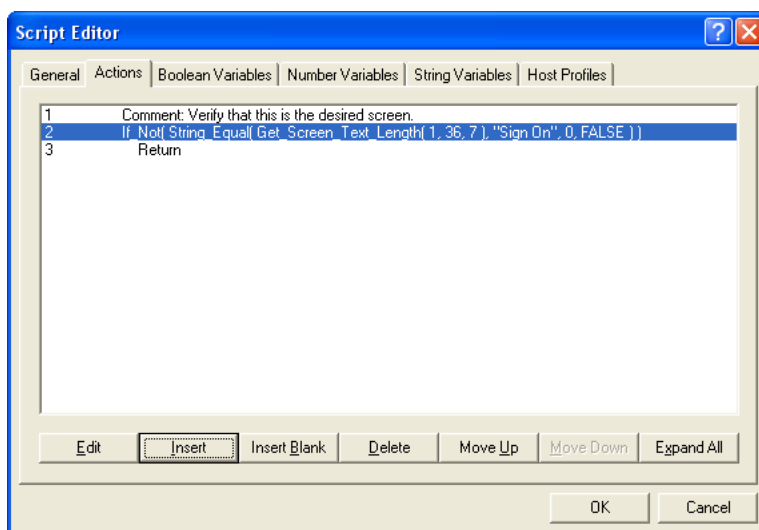


Figure 5-2. *Entering an IF_Not Action*

21 Click the **Insert** button

22 From the **Action** drop-down menu, select **Return**.

23 Click **OK**.

The code appears in the **Actions** tab.

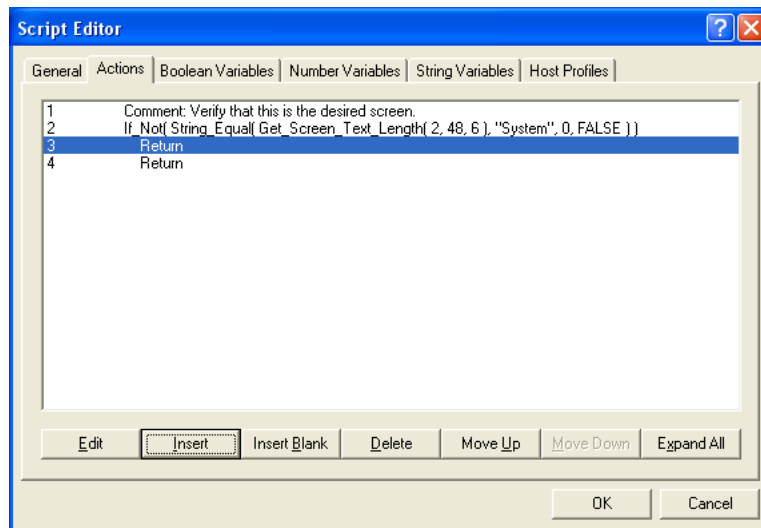


Figure 5-3. *Entering a Return*

24 Click the **Insert** button.

25 From the **Action** drop-down menu, select **End_If**.

26 Click **OK**.

The code appears in the **Actions** tab.

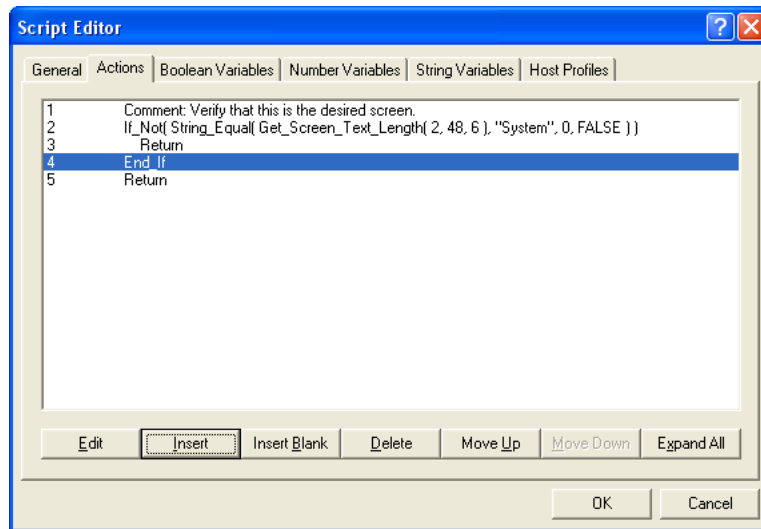


Figure 5-4. *Entering the End_If Action*

27 Click the **Insert** button.

28 From the **Action** drop-down menu, select **Set_Cursor_Position**.

29 Click the **Row** tab and enter **6** in the **Constant Number** text box.

30 Click the **Column** tab and enter **53** in the **Constant Number** text box.

31 Click **OK**.

The code appears in the **Actions** tab.

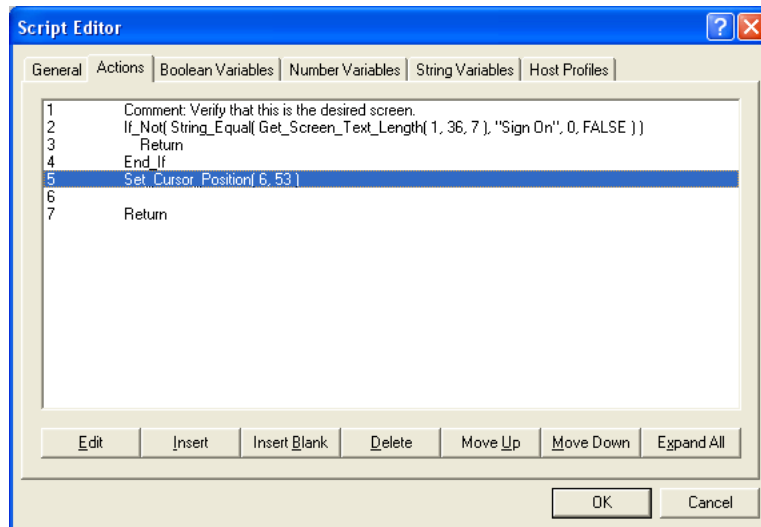


Figure 5-5. *Entering the Set_Cursor_Position Action*

32 From the **Action** tab, click the **Insert Blank** button to insert a blank line in the code.

33 Click the **Insert** button.

34 From the **Action** drop-down menu, select **Message**.

35 Click the **Message** tab and enter `Starting Script` in the **Constant Number** text box.

This code displays a “Starting Script” message on the emulation screen.

36 Click the **Timeout (Seconds)** tab and enter the number `3` in the **Constant Number** text box.

37 Click **OK**.

The code appears in the **Actions** tab.

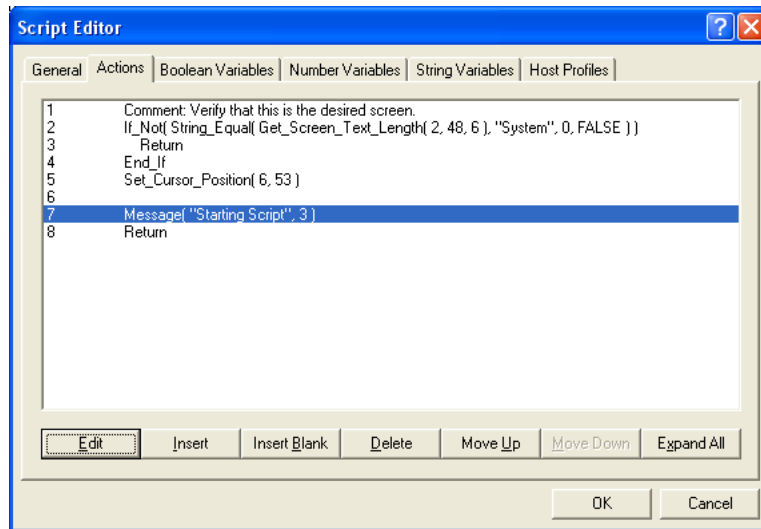


Figure 5-6. *Entering the Message Code*

Entering the User Name and Password

This portion of the script enters the login information.

To build the script code:

- 1 Click the **Insert Blank** button to insert a blank line in the code.
- 2 Click the **Insert** button.
- 3 From the **Action** drop-down menu, select **Keypress_String**.
- 4 Click the **Characters** tab and enter `User Name` in the **Constant String** text box.
- 5 Click **OK**.

The code appears in the **Actions** tab.

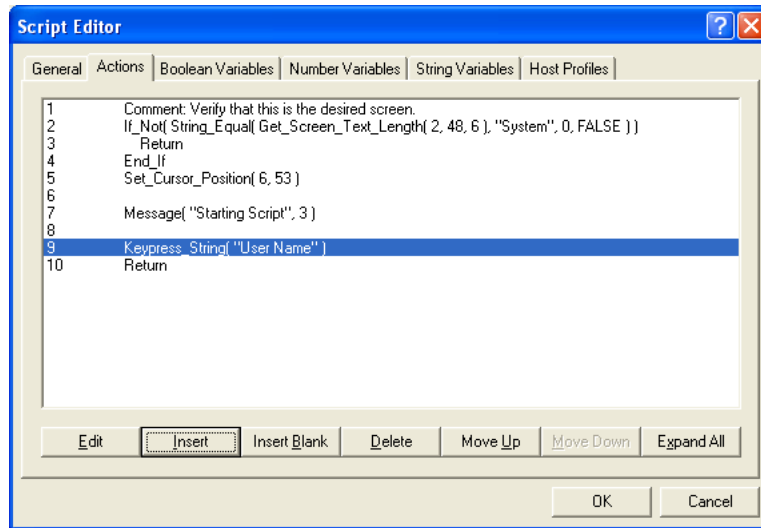


Figure 5-7. *Entering the User Name Code*

- 6** Click the **Insert** button.
- 7** From the **Action** drop-down menu, select **Keypress_Key**.
- 8** Click the **Key** tab.
- 9** From the **Emulation** drop-down menu, select the emulation type.
- 10** From the **Key** drop-down menu, select **Down Arrow**.
- 11** Click **OK**.

The code appears in the **Actions** tab.

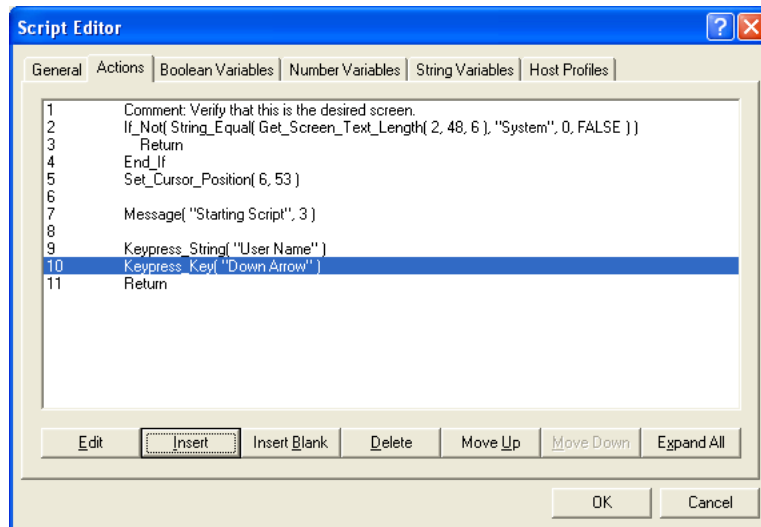


Figure 5-8. *Entering the Down Arrow Keypress*

- 12** Click the **Insert** button.
- 13** From the **Action** drop-down menu, select **Keypress String**.
- 14** Click the **Character** tab and enter `Password` in the **Constant String** text box.
- 15** Click **OK**.

The code appears in the **Actions** tab.

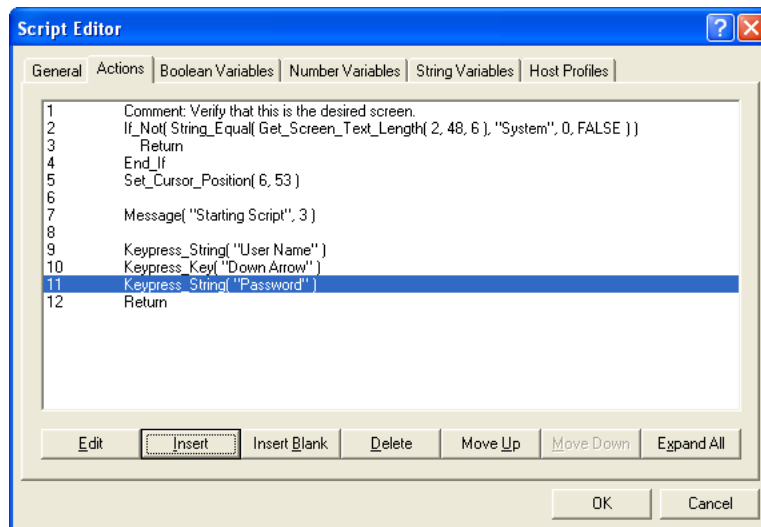


Figure 5-9. *Entering the Password Action*

- 16** Click the **Insert** button.
- 17** From the **Action** drop-down menu, select **Keypress_Key**.
- 18** Click the **Key** tab.
- 19** From the **Emulation** drop-down menu, select your emulation type.
- 20** From the **Key** drop-down menu, select **Enter**.
- 21** Click **OK**.

The code appears in the **Actions** tab.

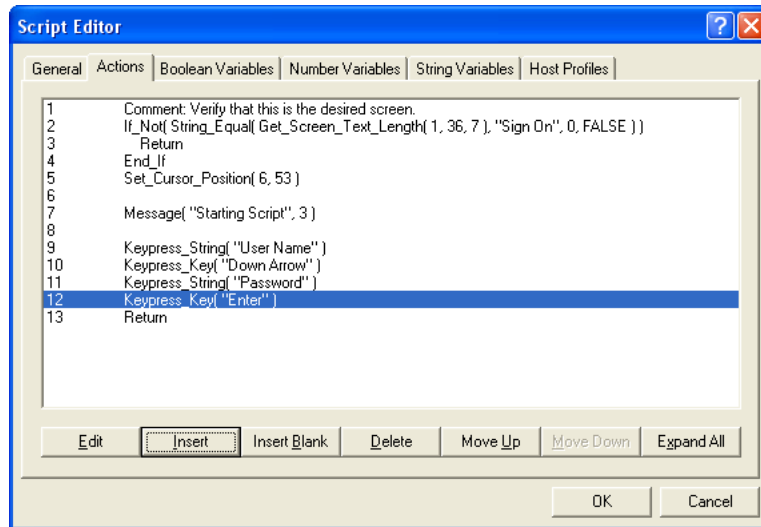


Figure 5-10. *Entering an Enter Action*

Verifying the Screen and Navigating Menus

This portion of the code verifies that you are on the correct screen after login. If you are not on the correct screen, the script will wait until the correct screen appears. Once the you are on the correct screen, the script navigates to a specific menu. The script displays the message "Script Done" and the script exits.

To build the script code:

- 1 Click the **Insert Blank** button.
- 2 Click the **Insert** button.
- 3 From the **Action** drop-down menu, select **Comment**.
- 4 Click the **Comment** tab and enter Wait for desired screen in the **Constant String** text box.
- 5 Click **OK**.

The code appears in the **Actions** tab.

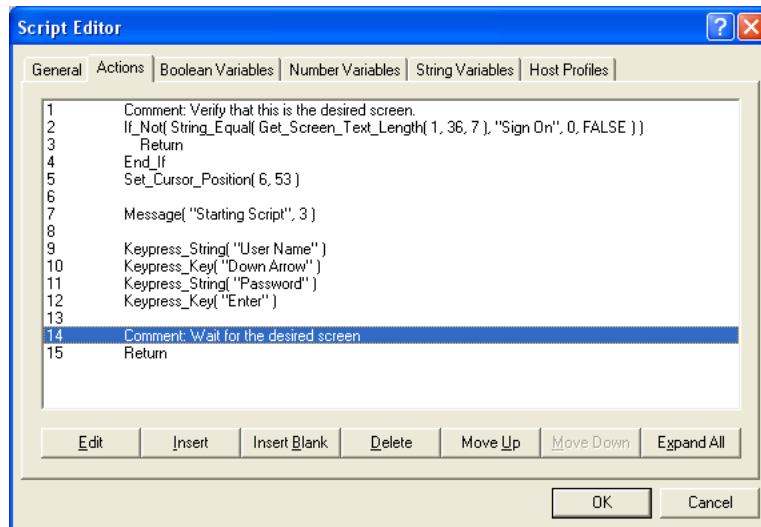


Figure 5-11. *Entering the Delay Action*

- 6** Click **Insert**.
- 7** From the **Action** drop-down menu, select **While_Not**.
- 8** Click the **Test** tab.
- 9** From the **Action** drop-down menu, select **String_Equal**.
- 10** Click the **Edit Action Value** button.
- 11** Click the **Test 1** tab.
- 12** From the **Action** drop-down menu, select **Get_String_Text_Length**.
- 13** Click the **Test 2** tab and enter `9 . FUNCTION KEYS` in the **Constant String** text box.
- 14** Click the **Maximum Length** tab and enter the number `0` in the **Constant Number** text box.
- 15** Click the **Ignore Case** tab and enable the **FALSE** option.
- 16** Click the **Test 1** tab.
- 17** Click the **Edit Action Value** button.

- 18 Click the **Row** tab and enter the number 11 in the **Constant Number** text box.
- 19 Click the **Column** tab and enter the number 1 in the **Constant Number** text box.
- 20 Click the **Maximum Length** tab and enter the number 16 in the **Constant Number** text box.
- 21 Click **OK** until you return to the **Action** tab.

The code is added to the **Action** tab.

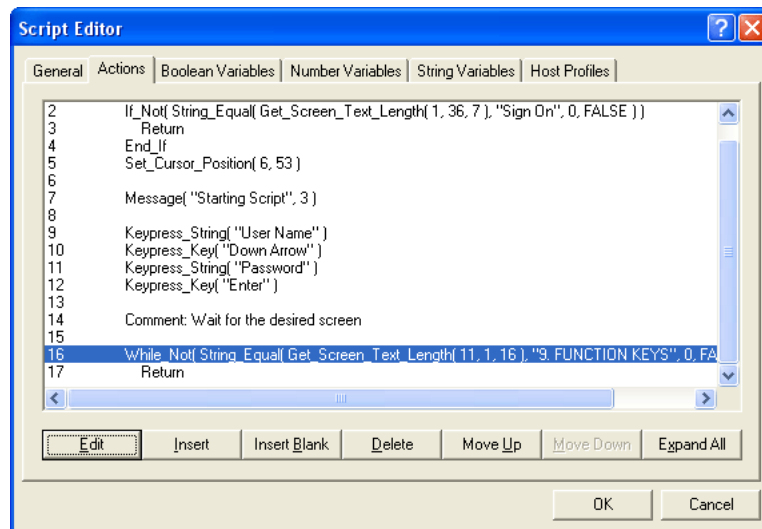


Figure 5-12. *Entering a While_Not Statement*

- 22 Click the **Insert** button.
- 23 From the **Action** drop-down menu, select **Wait_For_Screen_Update**.
- 24 Click **OK**.

The code appears in the **Actions** tab.

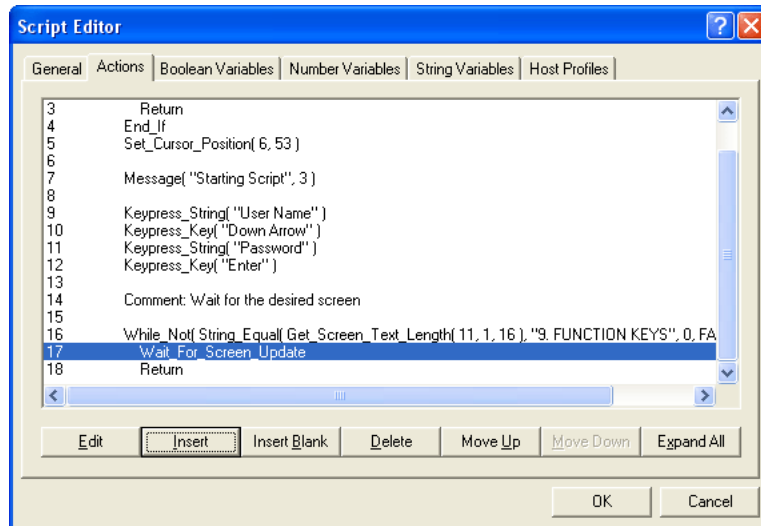


Figure 5-13. *Entering a Wait_For_Screen_Update Action*

25 Click the **Insert** button.

26 From the **Action** drop-down menu, select **End_While**.

27 Click **OK**.

The code appears in the **Actions** tab.

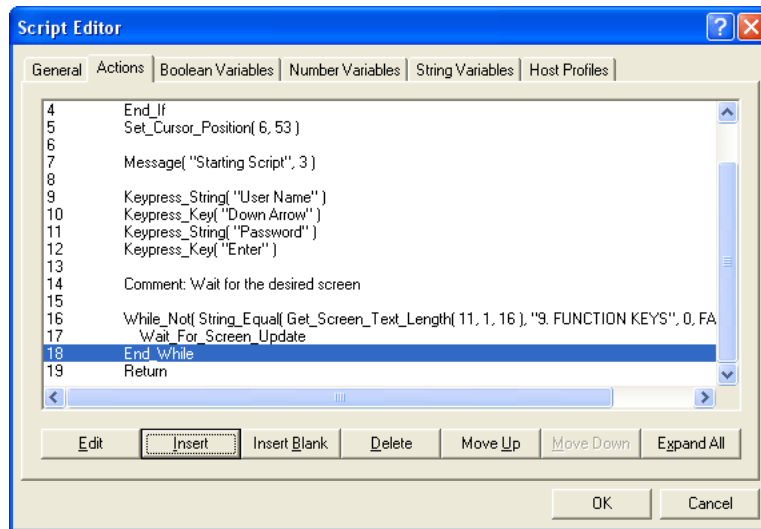


Figure 5-14. *Entering an End_While Action*

- 28** Click the **Insert Blank** button to insert a blank line in the code.
- 29** Click the **Insert** button.
- 30** From the **Action** drop-down menu, select **Keypress_String**.
- 31** Click the **Characters** tab and enter the number **9** in the **Constant String** text box.
- 32** Click **OK**.

The code is added to the **Actions** tab.

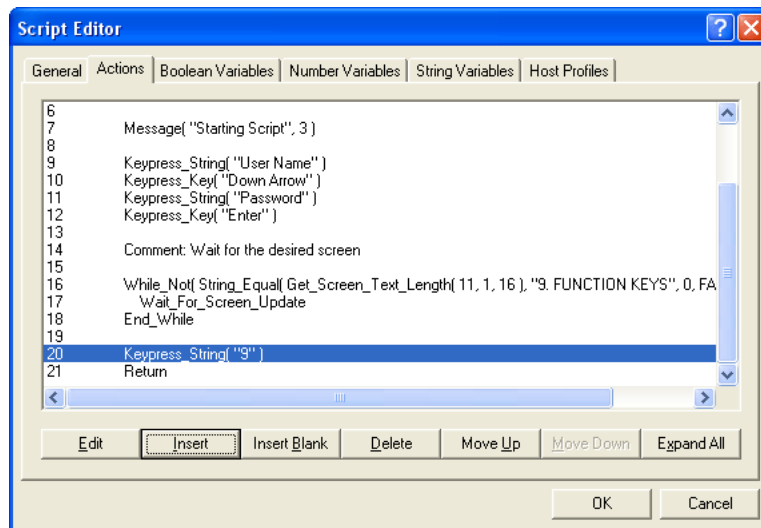


Figure 5-15. *Entering a Keypress Action*

33 Click the **Insert Blank** to insert a blank line in the code.

34 Click the **Insert** button.

35 From the **Action** drop-down menu, select **Message**.

36 Click the **Message** tab and enter `Script Done` in the **Constant String** text box.

37 Click the **Time(Milliseconds)** tab and enter the number `3` in the **Constant Number** text box.

38 Click **OK**.

The code is added to the **Actions** tab.

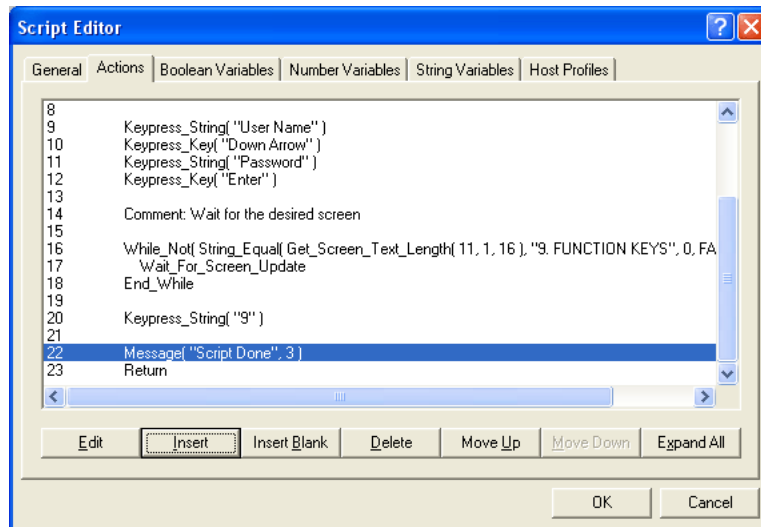


Figure 5-16. *Entering a Message Action*

The code is complete.

39 Click **OK** to save the code in the Script Editor script list.

Once you complete the script you have two options:

- Export the script to a specified location. Refer to *Saving and Exporting Scripts* on page 37 for more information.
- Execute the script by launching the Telnet Client and performing the activation method you assigned to this script. Refer to *Chapter 4: Executing Scripts* on page 47 for more information.

Appendix A: Examples

This appendix provides example scripts. Use these script examples to customize your own scripts according to preference.

Example 1: Beep

This is an example of a script that tells the device to beep if the word `ALARM` appears on the top five rows of the screen.

Example Code

```
If_Not(Search_Screen("ALARM",1,5,FALSE))
Return
End_If
Beep(1000,200,5)
Delay(200)
Beep(1500,500,9)
Return
```

Notes

This example should be set to activate each time the screen changes. Make sure that the `ALARM` text disappears quickly after being shown. Otherwise, the alarm will go off each time the screen updates (because the user pressed a key, each character from a bar code scanned was shown on the screen, etc.).

Here is an alternate implementation that waits for the `ALARM` text to disappear. The limitation with this version is that no other scripts will be able to run until the `ALARM` text is removed from the screen.

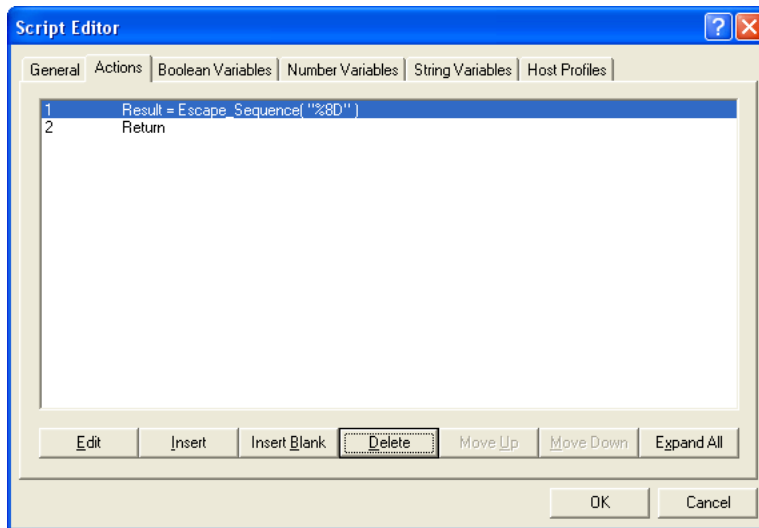
```
If_Not(Search_Screen("ALARM",1,5,FALSE))
Return
End_If
Beep(1000,200,5)
Delay(200)
Beep(1500,500,9)
While(Search_Screen("ALARM",1,5,FALSE))
Wait_For_Screen_Update
End_While
Return
```

Example 2: Escape Sequence

This is an example of using an escape sequence to turn off the Codabar symbology.

Example Code

```
Result=Escape_Sequence ("%8D")  
Return
```



Creating an Escape Sequence Script

Notes

You need to create a string variable named `Result` for this example, since the `Escape_Sequence` command returns a string value.

Refer to *Creating Variables* on page 27 for more information on creating variables.

Example 3: Request Information

This example requests the mobile device user to input some information and displays the result.

Example Code

```

    Result=Ask_String("Enter a string:", "Length
Calculator", 0, 200, "")
    Ask_OK(String_Combine("The string", String_Combine(
Result, String_Combine("is", String_Combine(
Number_To_String_Decimal(String_Length(Result)), "characte
rs long."))))), "String Length")
    Return

```

Notes

This example also requires a string variable named `Result`. The `ASK_OK` instruction uses actions inside of actions to get several layers deep. You could also use variables to break that instruction into several short instructions.

Example 4: Display Screen Button

This example displays a log out button that appears in the in the bottom-left corner of the screen (row two, column five) when the text “logged out” appears on the screen. Pressing the button allows you to exit Telnet.

Example Code

```

    If_Not(String_Equal( Get_Screen_Text_Columns(2,5,10),
"logged out", 0, FALSE ) )
        Return
    End_If
    Button_Create_View( "Exit", 1000,1,0,ButtonPressed)
    While_Not( ButtonPressed)
        If_Not(String_Equal(Get_Screen_Text_Columns(2,5,10),
"logged out", 0, FALSE))
            Return
        End_If
        Wait_For_Screen_Update
    End_While
    Exit_Application(0)

```

Notes

This example uses a boolean variable “`ButtonPressed`” to know if the screen button is pressed. The button is destroyed when the script exits so you do not need to delete the script.

The `While_Not` loop uses the `Wait_For_Screen_Update` action to detect if the `logged out` text is no longer there so that the scripts do not continually loop.

Sample Voice-Enabled Emulation Scripts

This section contains example scripts that perform various Voice-Enabled Emulation functions. For information on using the sample voice scripts, refer to *Wavelink Voice-Enabled Emulation User Guide*.

Play_Screen Sample Script

The following example script converts the current Telnet Client screen into speech that the user can hear.

```
nNumRows=Get_Screen_Rows
nCurrentRow=1
While (Number_Less_Than_Or_Equal(nCurrentRow,nNumRows))
    Speech_From_Text(Get_Screen_Text
        (nCurrentRow,1),FALSE)
    nCurrentRow=Number_Plus(nCurrentRow,1)
End_While
Return
```

Get_Number_Test Sample Script

The following example script converts a spoken number into text that displays on the mobile device. This script must be used in conjunction with the `Get_Number` sample script.

```
Speech_From_Text("Say a number",FALSE)
Call:Get_Number
    nResultt <--> nResult
Ask_OK(Number_To_String_Decimal(nResult),
    "Number Returned")
Return
```

Get_Number Sample Script

The following example script is called by the `Get_Number_Test` script. It retrieves the appropriate number for the `Get_Number_Test` script to display.

```
Comment:This script is designed to be called by other
scripts.
Comment:The result of the Speech-to-Text will be in the
nResult variable.
```

Comment: The number.bnf file must be available as a grammar file.

```
Speech_To_Text(sResult,"number")
nResult=0
While_Not(String_Empty(sResult))
  nNextSpace=String_Find_First(sResult,"",FALSE)
  nResult=Number_Plus(nResult,String_To_Number_Decimal
    (sResult))
  If_Number_Less_Than(nNextSpace,0))
    Break
  End_If
  nNextSpace=Number_Plus(nNextSpace,1)
  sResult=String_Right(sResult,Number_Minus
    (String_Length(sResult),nNextSpace))
End_While
Return
```

Speech_Button_Demo Sample Script

The following example script creates the following buttons on the screen: Digits, State, Play Screen, Done. When selected, the buttons allow the user to verbally input data.

```
While_Not(bExit)

  If_Not(bButtonsVisible)
    Button_Create_View("Digits",999,1,6,bGetDigits)
    Button_Create_View("State",999,16,5,bGetState)
    Button_Create_View("PlayScreen",1000,1,11,
      bPlayScreen)
    Button_Create_View("Done",1000,13,4,bExit)
  End_If

  Wait_For_Screen_Update

  If(bPlayScreen)
    bPlayScreen=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE
    Delay(1)

    numRows=Get_Screen_Rows
    nCurrentRow=1
    While(Number_Less_Than_Or_Equal
      (nCurrentRow,numRows))
      Speech_From_Text(Get_Screen_Text
        (nCurrentRow,1),FALSE)
      nCurrentRow=Number_Plus(nCurrentRow,1)
    End_While
  End_If
```

```
If (bGetDigits)
    bGetDigits=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE

    Message("Say 1 or more digits...",0)
    szResult=""
    Speech_To_Text(szResult,"connected_digits")
    Message_Clear
    szResult=String_Strip_Characters(szResult,"",FALSE)
    Keypress_String(szResult)
End_If

If (bGetState)
    bGetState=FALSE
    Button_Remove_All
    bButtonsVisible=FALSE

    Message("Say a USA state...",0)
    szResult=""
    Speech_To_Text(szResult,"usa_states")
    Message_Clear
    Keypress_String(szResult)
End_If

End_While
Button_Remove_All
Return
```

Appendix B: Actions

This appendix describes each scripting action. The appendix is 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 actions that return no value. The following action categories are described in this section:

- [Blank Line and Comment Actions](#)
- [Goto Support Actions](#)
- [Macro Exiting](#)
- [Conditionals](#)
- [General Queries](#)
- [Send Characters](#)
- [Message](#)
- [Sounds](#)
- [Waiting](#)
- [Logging](#)
- [Call Other Macros](#)
- [Screen Buttons](#)

- [Reboot](#)

Blank Line and Comment Actions

Blank_Line

Proceeds to the next instruction without taking any action.

Comment

Proceeds to the next instruction without taking any action.

Goto Support Actions

Goto

Jumps to the supplied label.

Label

Label to which a `Goto` can jump.

Macro Exiting

Return

Exits the script normally.

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

Abort

Exits the script immediately.

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

Abort_All

Exits all scripts for the session

Disconnect

Exits all scripts for the session and disconnects the session.

Exit_Application

Shuts down the Telnet application

The Return Value is the application exit value Telnet will use when it exits.

Conditionals

If

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

If_Not

If the Test is FALSE, continues executing until the next `Else` or `EndIf` statement.

Otherwise, only executes actions (if any) between the next `Else` and `EndIf` statements.

Else

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

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

End_If

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

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.

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

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 proceeds to the next `EndWhile` statement.

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

End_While

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

Continue

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

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

Break

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

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

General Queries**Ask_OK**

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

Send Characters**Keypress_String**

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

Keypress_Key

Sends a single keypress to the Telnet session. This is useful for emulation keys that `Keypress_String` cannot handle.

Scan_String

Treats the string as scanned data of the type specified.

Set_Cursor_Position

Moves the cursor to the specified row and column.

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

Message**Message**

Displays the message on the Telnet screen.

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

Message_Clear

Clears the message on the Telnet screen.

Sounds

Beep

Causes the device to beep. A Frequency of 1000 is a good default.

The Duration is in milliseconds, so a value of 1000 would be 1 second.

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

Play_Sound

Causes the device to play the sound specified by the sound name. The sound name may be any `.wav` file located in the folder specified by the emulation parameters or Resource Editor.

Waiting

Wait_For_Screen_Update

Suspends the current script until the screen has been updated.

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

Delay

Suspends the current script until the specified time has passed.

The time is in milliseconds, so a value of 1000 would be 1 second.

Logging

Logging_On

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

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.

Logging_Off

Turns off logging for the script.

Call Other Macros

Call

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

Refer to [Script Nesting](#) on page 41 for more information.

Screen Buttons

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.

NOTE You can add bitmaps to the resource file by using the Resource Editor.

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.

Button_Bitmap_Create_View

This command is the same as `Button_Bitmap_Create_Emulation`, except that the screen position is used instead of the text position, allowing the button to always be visible. For example, 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.

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.

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.

Button_Create_View

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.

For example, 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, 990 would start 11 columns left of the upper-right corner of the screen.

Button_Remove

Removes a button created with the `Button_Create_Emulation` and `Button_Create_View` actions with the specified text.

Button_Remove_All

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

Reboot

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

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

NOTE Cold boot is only supported by some mobile devices.

Boolean Values

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

- [Boolean Assignments](#)
- [Boolean Comparisons](#)
- [String Comparisons](#)
- [Field Identifiers and Data](#)
- [Integer Comparison](#)
- [General Queries](#)
- [Suspend](#)
- [Search the Screen](#)
- [WEB Emulation Commands](#)
- [Speech Commands](#)

Boolean Assignments

Boolean_Set

Returns TRUE if the Test is TRUE, FALSE otherwise.

Boolean_Not

Returns FALSE if the Test is TRUE, TRUE otherwise.

Boolean_And

Returns TRUE if all test values are TRUE. Returns FALSE otherwise.

All tests will be evaluated each time this action is taken.

Boolean_Or

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

All tests will be evaluated each time this action is taken.

Boolean Comparisons**Boolean_Equal**

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

Boolean_Not_Equal

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

String Comparisons**String_Empty**

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

String_Less_Than

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

String_Less_Than_Or_Equal

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

String_Equal

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

String_Greater_Than_Or_Equal

Returns `TRUE` if `Test1` follows `Test2` in alphabetical ordering or they are the same string, `FALSE` otherwise.

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.

String_Greater_Than

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

String_Not_Equal

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

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.

Field Identifiers and Data

Set_Field_Data_ID

Sets the Data ID for a field. Returns `TRUE` if successful, `FALSE` if the field index is not valid.

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.

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

Set_Field_Symbology_ID

Sets the Symbology ID for a field. 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.

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.

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

Get_Field_Symbology_Operator

Returns 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.

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

Set_Field_Append_Scan_Data

Controls whether to append scan data in the field. Returns TRUE if successful, FALSE if the field index is not valid.

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

Set_Field_Com_Data_Field

Sets a field to be the Com Data Field for the screen. Returns TRUE if successful, FALSE if the index is not valid.

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

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

Set_Field_Prefix_Scan_Data

Sets the data prefixed to a field when the field is scanned. Returns TRUE if successful, FALSE if the field index is not valid.

Use a blank string to clear the prefix data.

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

Get_Field_Append_Scan_Data

Query whether data is appended when the field is scanned.

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

Integer Comparison**Number_Less_Than**

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

Number_Less_Than_Or_Equal

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

Number_Equal

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

Number_Greater_Than_Or_Equal

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

Number_Greater_Than

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

Number_Not_Equal

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

General Queries**Ask_OK_Cancel**

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

Returns TRUE if the user presses `OK`, FALSE if the user presses `Cancel`.

Ask_Yes_No

Displays the message in a box with a `Yes` and `No` button and waits until the user presses a button. Returns TRUE if the user presses `Yes`, FALSE if the user presses `No`.

Suspend

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

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.

Search the Screen

Search_Screen

Searches the screen for the supplied text. Returns `TRUE` if the text is found, `FALSE` otherwise.

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.

WEB Emulation Commands

Web_Navigate

Navigates WEB emulation to the URL provided. Returns `TRUE` if the navigation was successful; returns `FALSE` otherwise.

Web_Navigate_Frame

Navigates WEB emulation to the URL provided within the indicated frame (`\Frame Name\`). Returns `TRUE` if the navigation was successful; returns `FALSE` otherwise.

Web_Navigate_Post_Data

Navigates WEB emulation to the URL provided. The `\Post Data\` is sent to the server using an HTTP POST transaction (rather than an HTTP GET transaction). Returns `TRUE` if the navigation was successful; returns `FALSE` otherwise.

Web_Scripting

Instructs WEB emulation to execute the scripting information. The code should start with a `\javascript:\` or `\vbscript:\` string to ensure that the correct scripting type is used. Returns `TRUE` if the script execution started successfully; returns `FALSE` otherwise.

Web_Search_Source

Searches the page source of the current WEB emulation page. Returns TRUE if the text is found anywhere in the page source; returns FALSE otherwise. If `\Search Frames\` is TRUE, the page source of any frames will be searched as well.

Speech Commands**Speech_From_Text_Available**

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

Speech_From_Text

Converts text into sound and plays the resulting sound on the computer. Returns TRUE if the sound was played successfully; returns FALSE otherwise. If `\Wait Until Done\` is FALSE, the script will continue to execute while the sound is being played.

Speech_To_Text_Available

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

Speech_To_Text

Returns the text equivalent of a user's speech. Returns an empty string if no acceptable speech was detected. If a grammar is specified, the grammar file with that name is used for speech recognition; otherwise, the previous grammar file is reused.

Speech_To_Text_No_Wait

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

Speech_To_Text_Cancel

Returns after canceling the last `Speech_To_Text_No_Wait` action. Returns immediately if there is no action to cancel. Provides a way for the script to do other things while the Speech-to-Text action occurs.

Speech_Setting_Available

Identifies speech settings by case-insensitive name strings. Returns TRUE if the speech setting name is supported; returns FALSE otherwise. Refer to the *Wavelink Voice-Enabled Emulation User Guide* for a list of available setting names.

Speech_Change_Setting

Changes the speech setting to the specified value. Returns TRUE if the setting is supported and the value is valid for that setting; returns FALSE otherwise.

Speech_Get_Setting

Returns the current value for the speech setting. Returns -1 if the speech setting is not valid.

Speech_Get_Setting_Max

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.

Speech_Find_Setting_Value

Searches all possible value descriptions for the speech setting and 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.

Speech_Get_Setting_Value_Desc

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.

Speech_To_Text_Get_User_Name

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.

Speech_To_Text_Change_User_Name

Changes the user name being used by the speech-to-text engine. Returns TRUE if the user name was changed, FALSE if this feature is not supported.

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](#)
- [Scanner Information](#)
- [ESC Sequence Support](#)
- [String Variable Assignments](#)
- [Number to Character Conversion](#)
- [Field Identifiers and Data](#)

Get System Information

Get_MAC_Address

Returns the current MAC address for the device.

Get_IP_Address

Returns the current IP Address for the device.

Get_Field_Symbology_ID

Returns the field symbology ID. 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.

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

Get_Screen_Text

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

Get_Screen_Text_Length

Returns the text starting at the specified screen position up to the right side of the display. The string will be truncated if it is longer than the number of characters specified.

Get_Screen_Text_Columns

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

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

Get_Workstation_ID

Returns the current Workstation ID.

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.

Scanner Information**Get_Scan_Type_Name**

Returns the name of the supplied scan type.

An empty string is returned if the scan type is not recognized.

ESC Sequence Support**Escape_Sequence**

Handles the supplied Wavelink Custom or Telxon ESC Sequence for all emulation types. The sequence should be all the characters that will follow the first ESC character. The string returned will be the sequence returned by the ESC sequence (without the initial ESC) or an empty string if the sequence returns nothing.

String Variable Assignments**String_Set**

Returns the value of the string.

String_Combine

Returns the value of string1 concatenated with string2.

String_Left

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.

String_Right

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.

String_Middle

Returns a string with just the middle *n* characters of the input string.

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`.

If the input string is less than *n* characters, the entire string is returned.

String_Upper

Returns a string with all characters converted to uppercase.

String_Lower

Returns a string with all characters converted to lowercase.

String_Replace

Returns a string where all instances of `Substring to Replace` have been replaced with `Replacement Substring`. If `Ignore Case` is TRUE then upper-case and lower-case letters are considered to be equal.

String_Only_Chars

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

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

String_Strip_Chars

Returns a string where all characters in `String to Parse` that are in `Characters to Strip` have been deleted.

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

String_Trim_Spaces_Start

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

String_Trim_Spaces_End

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

Number_To_String_Binary

Returns a string with the binary (base 2) representation of the number.

Number_To_String_Octal

Returns a string with the octal (base 8) representation of the number.

Number_To_String_Decimal

Returns a string with the decimal (base 10) representation of the number.

Number_To_String_Hexadecimal_Lowercase

Returns a string with the hexadecimal (base 16) representation of the number using lowercase characters.

Number_To_String_Hexadecimal_Uppercase

Returns a string with the hexadecimal (base 16) representation of the number using uppercase characters.

Ask_String

Displays a dialog asking the user for a string, and returns the string supplied by the user. The supplied default string is returned (unaltered) if the user cancels the dialog.

Ask_String_Password

Displays a dialog asking the user for a string, and returns the string supplied by the user.

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

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

Ask_String_Uppercase

Displays a dialog asking the user for a string, and returns the string supplied by the user. Any lowercase letters entered are converted to uppercase characters. The supplied default string is returned (unaltered) if the user cancels the dialog.

Ask_String_Lowercase

Displays a dialog asking the user for a string, and returns the string supplied by the user. Any uppercase letters entered are converted to lowercase

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

Number to Character Conversion

Number_To_Character

Returns a string one character in length, where the value for that character is the supplied number. For example, a number value of 87 would return a string consisting of a `w` -- the ASCII character for value 87.

Field Identifiers and Data

Get_Field_Data_ID

Returns the field's Data ID. 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.

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

Get_Field_Prefix_Scan_Data

Returns the data prefixed when the field is scanned.

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

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](#)
- [Scanner Information](#)
- [General Queries](#)
- [String Handling](#)
- [Integer Assignments](#)
- [Convert Strings to Integers](#)
- [Ask User for Integer](#)

- [Number/Character Conversion](#)

Get System Information

Get_Screen_Columns

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

Get_Screen_Rows

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

Get_Position_Column

Get the column number on which the cursor is currently located. The left-most column is 1

Get_Position_Row

Get the row number on which the cursor is currently located. The top-most row is 1.

Get_Session_Number

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

Get_Time

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

Get_Time_Since_Reset

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

Get_Field_Index

Get the index of a field at the specified row and column. An index of -1 means there is no field at the row and column.

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

Get_Num_Fields

Get the number of fields on the screen.

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

Get_Field_Index_Row_Text

Get the index of a field that is in the same row as the text. 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 is only valid when using IBM 5250 or 5555 emulation.

Get_Field_Index_Column_Text

Get the index of a field that is in the same column as the text. The text may be above or below the field, in the same column as the field. An index of -1 means either the text was not found or there is no field above or below the text in the column where the text was found.

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

Get_Field_Row

Get the row number of the field. A row number of 0 means that the field index is not valid.

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

Get_Field_Column

Get the column number of the field. A column number of 0 means that the field index is not valid.

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

Get_Field_Length

Get the length of the field. A length of 0 means that the field index is not valid.

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

Get_Num_Field_Data_IDs

Get the number of data IDs in a field. The number -1 means the field index is not valid.

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

Get_Num_Field_Symbology_IDs

Get the number of symbology IDs in a field. The number -1 means the field index is not valid.

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

Get_Field_Com_Data_Field

Get the index of the field that is the Com Data Field. An index of -1 means that no field is the Com Data Field.

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

Scanner Information**Get_Scan_Type_Value**

Returns the value of the supplied scan type name. A value of 0 is returned if the scan type name is not recognized.

General Queries**Ask_Yes_No_Cancel**

Displays the message in a box with a **Yes**, **No** and **Cancel** button, and waits until the user presses a button. If the `Make No Default` value is `TRUE`, then the **No** button is the default. Otherwise, the **Yes** button is the default.

Returns 2 if the user presses **Yes**, 1 if the user presses **No**, and 0 if the user presses **Cancel**.

String Handling**String_Length**

Returns the number of characters in the string.

Returns 0 if the string is empty (has no characters).

String_Find_First

Finds the first instance of the substring inside the string, and returns the position where that substring starts. 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.

String_Find_Last

Finds the last instance of the substring inside the string, and returns the position where that substring starts. The left-most position is 0, so a value of 0 would be returned if the only substring found was at the beginning of the string. A value of -1 is returned if no instances of the substring are in the string.

Integer Assignments

Number_Set

Returns the value of the number.

Number_Plus

Returns the sum of the two numbers.

Number_Minus

Returns the value when Number2 is subtracted from Number1.

Number_Multiply

Returns the product of the two numbers.

Number_Divide

Returns the value when Number1 is divided by Number2.

Because the numbers are integers, the remainder is ignored. For example, 7 divided by 3 would return 2.

Number_Divide_Remainder

Returns the remainder when Number1 is divided by Number2. For example, 7 divided by 3 would return a remainder of 1.

Convert Strings to Integers

String_To_Number_Binary

Returns the binary (base-2) number represented by the string. Parsing the string continues until a character other than a 0 or 1 is reached.

If the string does not represent a binary number, a 0 is returned.

String_To_Number_Octal

Returns the octal (base-8) number represented by the string. Parsing the string continues until a character other than a 0 through 7 is reached. If the string does not represent an octal number, a zero is returned.

String_To_Number_Decimal

Returns the decimal (base-10) number represented by the string. Parsing the string continues until a character other than a 0 through 9 is reached. If the string does not represent a decimal number, a 0 is returned.

String_To_Number_Hexadecimal

Returns the hexadecimal (base-16) number represented by the string. Parsing the string continues until a character other than a 0 through 9, a through f, or A through F is reached. If the string does not represent a hexadecimal number, a 0 is returned.

Ask User for Integer**Ask_Number**

Displays a dialog asking the user for a decimal number, and returns the number supplied by the user. The supplied default value is returned if the user cancels the dialog.

Number/Character Conversion**Character_To_Number**

Converts the character at position Index in the string into the number value for that character. An index of 0 indicates the left-most character in the string. If the Index does not point to a character, a value of 0 is returned.

Appendix C: Symbologies and Values

The following is a list of symbologies and their values:

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

MAXICODE = 72

DATAMATRIX = 73

QR CODE = 74

MACRO MICROPDF = 75

RSS 14 = 76

RSS LIMITED = 77

RSS EXPANDED = 78

SIGNATURE = 82

WEBCODE = 84

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

POSICODE = 171

UPC = 241

MSR = 245

RFID = 246

Appendix D: Wavelink Contact Information

If you have comments or questions regarding this product, please contact Wavelink Customer Service via email or telephone.

Email: customerservice@wavelink.com

Phone: 1-888-699-WAVE(9283)

Index

A

- Abort 78
- Abort_All 78
- about scripting 10
- actions 77
- activation method
 - on barcode, MSR or RFID Scan 23
 - on key combination 21
 - on screen update 25
 - select from menu 20
 - when session connects 22
- Ask User for Integer
 - Ask_Number 101
- Ask_Number 101
- Ask_OK 80
- Ask_OK_Cancel 88
- Ask_String 95
- Ask_String_Lowercase 95
- Ask_String_Password 95
- Ask_String_Uppercase 95
- Ask_Yes_No 88
- Ask_Yes_No_Cancel 99

B

- Beep 81
- beep 81
- Blank Line and Comment Actions
 - Blank_Line 78
 - Comment 78
- Blank_Line 78
- Boolean Assignments
 - Boolean_And 84
 - Boolean_Not 84
 - Boolean_Or 85
 - Boolean_Set 84
 - Field Identifiers and Data 86
- Boolean Comparisons
 - Boolean_Equal 85

- Boolean_Not_Equal 85
- Boolean General Queries
 - Ask_OK_Cancel 88
 - Ask_Yes_No 88
- Boolean Integer Comparison
 - Number_Equal 88
 - Number_Great_Than 88
 - Number_Greater_Than_Or_Equal 88
 - Number_Less_Than 88
 - Number_Less_Than_Or_Equal 88
 - Number_Not_Equal 88
- Boolean String Comparisons
 - String_Empty 85
 - String_Equal 85
 - String_Greater_Than 86
 - String_Greater_Than_Or_Equal 86
 - String_Less_Than 85
 - String_Less_Than_Or_Equal 85
 - String_Not_Equal 86
- Boolean Values
 - Boolean Assignments 84
 - Boolean Comparisons 85
 - General Queries 88
 - Search the Screen 89
 - Speech 90
 - String Comparisons 85
 - Suspend 88
 - WEB Emulation 89
- Boolean_And 84
- Boolean_Equal 85
- Boolean_Not 84
- Boolean_Not_Equal 85
- Boolean_Or 85
- Boolean_Set 84
- Break 80
- Button_Bitmap_Create_Emulation 82
- Button_Bitmap_Create_View 82

- Button_Create_Emulation 83
- Button_Create_View 83
- Button_Remove 83
- Button_Remove_All 83

C

- Call 82
- Call Other Macros
 - Call 82
- Comment 78
- Conditionals
 - Break 80
 - Continue 80
 - Else 79
 - End_If 79
 - End_While 80
 - If 79
 - If_Not 79
 - While 79
 - While_Not 79
- configuration overview 18
- considerations 11
- contact information 107
- Continue 80
- Convert Strings to Integers
 - String_To_Number_Binary 100
 - String_To_Number_Decimal 100
 - String_To_Number_Octal 100
 - String_To_Number-Hexadecimal 101
- creating
 - log files 39
 - script code 27
 - scripts 17
 - scripts from text 33
 - scripts manually 18

D

- deploying scripts 38
- Disconnect 78
- document

- assumptions 9
- conventions 10

E

- editing scripts 35
- Else 79
- End_If 79
- End_While 80
- Escape_Sequence 93
- examples 71
 - beep 71
 - building example script 51
 - display screen button 73
 - escape sequence 72
 - request information 72
- executing scripts 47
 - on barcode, MSR or RFID scan 49
 - on key combination 48
 - on screen update 49
 - select from menu 47
 - when session connects 48
- Exit_Application 78
- exporting 37
- exporting scripts 37

F

- Field Data ID 42
- Field Identifiers and Data
 - Get_Field_Append_Scan_Data 88
 - Get_Field_Data_ID 96
 - Get_Field_Prefix_Scan_Data 96
 - Get_Field_Symbology_Operator 87
 - Set_Field_Append_Scan_Data 87
 - Set_Field_Com_Data_Field 87
 - Set_Field_Data_ID 86
 - Set_Field_Prefix_Scan_Data 87
 - Set_Field_Symbology_ID 87

G

- General Queries

- Ask_OK 80
- Get_Field_Append_Scan_Data 88
- Get_Field_Column 98
- Get_Field_Com_Data_Field 99
- Get_Field_Data_ID 96
- Get_Field_Index 97
- Get_Field_Index_Column_Test 98
- Get_Field_Index_Row_Text 98
- Get_Field_Length 98
- Get_Field_Prefix_Scan_Data 96
- Get_Field_Row 98
- Get_Field_Symbology_ID 92
- Get_Field_Symbology_Operator 87
- Get_IP_Address 92
- Get_MAC_Address 92
- Get_Num_Field_Data_IDs 98
- Get_Num_Field_Symbology_IDs 98
- Get_Num_Fields 97
- Get_Position_Column 97
- Get_Position_Row 97
- Get_Screen_Columns 97
- Get_Screen_Rows 97
- Get_Screen_Text 92
- Get_Screen_Text_Columns 92
- Get_Screen_Text_Length 92
- Get_Session_Number 97
- Get_Time 97
- Get_Time_Since_Reset 97
- Get_Workstation_ID 93
- Goto 78
- Goto Support Actions 78
 - Goto 78
 - Label 78

H

- host profiles 29

I

- If 79
- If_Not 79

- importing
 - scripts 36
 - text file 33
- Importing Scripts 36
- Integer Assignments
 - Number_Device Remainder 100
 - Number_Divide 100
 - Number_Minus 100
 - Number_Multiply 100
 - Number_Plus 100
 - Number_Set 100
- Integer Character_To_Number 101
- Integer Comparison 88
- Integer General Queries
 - Ask_Yes_No_Cancel 99
- Integer Get System
 - Get_Field_Column 98
 - Get_Field_Com_Data_Field 99
 - Get_Field_Index 97
 - Get_Field_Index_Column_Text 98
 - Get_Field_Index_Row_Text 98
 - Get_Field_Length 98
 - Get_Field_Row 98
 - Get_Num_Field_Data_IDs 98
 - Get_Num_Field_Symbology_IDs 98
 - Get_Num_Fields 97
- Integer Get System Information
 - Get_Position_Column 97
 - Get_Position_Row 97
 - Get_Screen_Columns 97
 - Get_Screen_Rows 97
 - Get_Session_Number 97
 - Get_Time 97
 - Get_Time_Since_Reset 97
- Integer String Handling
 - String Handling 99
 - String_Find_First 99
 - String_Find_Last 99
 - String_Length 99
- Integer Values 96

- Convert Strings to Integers 100
- General Queries 99
- Get System Information 97
- Integer Assignments 100
- Number/Character Conversion 101
- Scanner Information 99

K

- Keypress_Key 80
- Keypress_String 80

L

- Label 78
- launching
 - from Avalanche Manager 13
 - from Avalanche MC 14
 - Script Editor 13
- log file 39
 - logging_off action 40
 - logging_on action 39
- Logging
 - Logging_Off 82
 - Logging_On 81
- Logging_Off 82
- Logging_On 81

M

- Macro Editing
 - Return 78
- Macro Exiting
 - Abort 78
 - Abort_All 78
 - Disconnect 78
 - Exit_Application 78
- Message
 - Message 80
 - Message_Clear 81
- Message_Clear 81

N

- naming scripts 19

- No Return Value 77, 78

- Blank Line and Comment Actions 78
- Call Other Macros 82
- Conditionals 79
- General Queries 80
- Logging 81
- Macro Exiting 78
- Message 80
- Reboot 83
- Screen Buttons 82
- Send Characters 80
- Sounds 81
- Waiting 81

- Number/Character Conversion 101

- Number_Divide 100

- Number_Divide_Remainder 100

- Number_Equal 88

- Number_Greater_Than 88

- Number_Greater_Than_Or_Equal 88

- Number_Less_Than 88

- Number_Less_Than_Or_Equal 88

- Number_Minus 100

- Number_Multiply 100

- Number_Not_Equal 88

- Number_Plus 100

- Number_Set 100

- Number_To_String_Binary 95

- Number_To_String_Decimal 95

- Number_To_String_Hexadecimal_Lowercase 95

- Number_To_String_Hexadecimal_Uppercase 95

- Number_To_String_Octal 95

O

- overview 11

P

- performing script capturing 31

- persistent variables 29

Play_Sound 81

R

Return 78

S

saving scripts 37

Scan_String 80

Scanner Information

Integer Get_Scan_Type_Value 99

String Get_Scan_Type_Name 93

Screen Buttons

Button_Bitmap_Create_Emulation 82

Button_Bitmap_Create_View 82

Button_Create_Emulation 83

Button_Create_View 83

Button_Remove 83

Button_Remove_All 83

script capturing 31

script code 27

script nesting 41

scripting

about 10

activation methods 19

design considerations 11

naming 19

overview 11

scripts 37

building an example 51

creating 17

deploying 38

editing 35

executing 47

importing 36

nesting 41

saving 37

syncing 38

scripts from text 33

Search_Screen 89

selecting

host profiles 29

Send Character

Keypress_String 80

Send Characters

Keypress_Key 80

Scan_String 80

Set_Cursor_Position 80

Set_Cursor_Position 80

Set_Field_Append_Scan_Data 87

Set_Field_Com_Data_Field 87

Set_Field_Data_ID 86

Set_Field_Prefix_Scan_Data 87

Set_Field_Symbology_ID 87

Sounds

Beep 81

Play_Sound 81

speech commands 90

Speech_Change_Setting 91

Speech_Find_Setting_Value 91

Speech_From_Text 90

Speech_From_Text_Available 90

Speech_Get_Setting 91

Speech_Get_Setting_Max 91

Speech_Get_Setting_Value_Desc 91

Speech_Setting_Available 91

Speech_To_Text 90

Speech_To_Text_Available 90

Speech_To_Text_Cancel 90

Speech_To_Text_Change_User_Name 91

Speech_To_Text_No_Wait 90

Speech_Change_Setting 91

Speech_Find_Setting_Value 91

Speech_From_Text 90

Speech_From_Text_Available 90

Speech_Get_Setting 91

Speech_Get_Setting_Max 91

Speech_Get_Setting_Value_Desc 91

Speech_Setting_Available 91

Speech_To_Text 90

Speech_To_Text_Available 90

- Speech_To_Text_Cancel 90
 - Speech_To_Text_Get_User_Name 91
 - Speech_To_Text_No_Wait 90
 - String Handling 99
 - String Values
 - ESC Sequence Support 93
 - Field Identifiers and Data 96
 - Get System Information 92
 - Number to Character Conversion 96
 - Scanner Information 93
 - String Variable Assignments 93
 - String Variable Assignments
 - Ask_String 95
 - Ask_String_Lowercase 95
 - Ask_String_Password 95
 - Ask_String_Uppercase 95
 - Number_To_String_Binary 95
 - Number_To_String_Decimal 95
 - Number_To_String_Hexadecimal_Lowercase 95
 - Number_To_String_Hexadecimal_Uppercase 95
 - Number_To_String_Octal 95
 - String_Combine 93
 - String_Left 93
 - String_Lower 94
 - String_Middle 94
 - String_Only_Characters 94
 - String_Replace 94
 - String_Right 93
 - String_Set 93
 - String_Strip_Characters 94
 - String_Trim_Spaces_End 94
 - String_Trim_Spaces_Start 94
 - String_Upper 94
 - String_Combine 93
 - String_Empty 85
 - String_Equal 85
 - String_Find_First 99
 - String_Find_Last 99
 - String_Greater_Than 86
 - String_Greater_Than_Or_Equal 86
 - String_Left 93
 - String_Length 99
 - String_Less_Than 85
 - String_Less_Than_Or_Equal 85
 - String_Lower 94
 - String_Middle 94
 - String_Not_Equal 86
 - String_Only_Characters 94
 - String_Replace 94
 - String_Right 93
 - String_Set 93
 - String_Strip_Characters 94
 - String_To_Number_Binary 100
 - String_To_Number_Decimal 100
 - String_To_Number_Hexadecimal 101
 - String_To_Number_Octal 100
 - String_Trim_Spaces_End 94
 - String_Trim_Spaces_Start 94
 - String_Upper 94
 - Strings Get System Information
 - Get_Field_Symbology_ID 92
 - Get_IP_Address 92
 - Get_MAC_Address 92
 - Get_Screen_Text 92
 - Get_Screen_Text_Columns 92
 - Get_Screen_Text_Length 92
 - Get_Workstation_ID 93
 - Symbologies and Values 103
 - syncing scripts 38
- T**
- Text Editor 34
 - text file, importing 33
- V**
- variables 27

W

Wait_For_Screen_Update 81
Waiting
 Delay 81
 Wait_For_Screen_Update 81
Wavelink contact information 107
WEB Emulation 89
 Web_Navigate 89
 Web_Navigate_Frame 89
 Web_Navigate_Post_Data 89
 Web_Scripting 89
 Web_Search_Source 90
Web_Navigate 89
Web_Navigate_Frame 89
Web_Navigate_Post_Data 89
Web_Scripting 89
Web_Search_Source 90
While 79
While_Not 79

