Speakeasy

Wavelink Speakeasy Reference Guide

Version 2.60

Revised 11/14/2014

Email: customerservice@wavelink.com
Website: http:\\www.wavelink.com

Table of Contents

Introduction to Speakeasy

Wavelink Speakeasy is a verbal communication system that facilitates real-time voice communication between the host computer and the mobile device user. It is available to use with the Wavelink Terminal Emulation (TE) Client 7.0 and newer versions on Windows Mobile or Windows CE devices. Speakeasy can be implemented from the Screen Reformatter or you can create scripts to customize how Speakeasy is used.

Speakeasy provides the ability to translate data from the host computer into spoken directions that the user is able to hear (text-to-speech). The user's spoken response can then be transcribed and sent back to the host computer (speech-to-text). You can install both text-to-speech and speech-to-text, or just the component that fits your company's needs. This introduction provides a list of Speakeasy features and an overview of implementing Speakeasy.

Speakeasy Features

Speakeasy has a variety of features that make using voice with terminal emulation easy and practical:

- **Client-side processing**. All Speakeasy processing is performed on the mobile device, reducing bandwidth usage and eliminating the need for middleware servers and host modifications.
- **Powerful scripting engine**. The Wavelink TE Client includes a scripting utility which can capture and manipulate screen data, launch secondary processes, provide audible warning and screen messages, and automate repetitive tasks and data entry. Scripts can be triggered by screen data, unique key combinations, or can be called from the reformatter.
- **WYSIWYG configuration with the Screen Reformatter.** Speakeasy options are available in the Terminal Emulation Screen Reformatter, making it easy to add speech-to-text and text-to-speech to screens. Use the Screen Reformatter to change Speakeasy settings, select the grammar files available, or perform specific functions for a speech-to-text result.
- **Microphone calibration tools.** Speakeasy has built-in calibration tools that use Speakeasy settings to improve microphone performance. There is a quick calibration that uses a few verbal prompts and responses for calibration, or a full calibration that provides tools for the user to change settings manually.
- **Wide range of available languages**. Speakeasy works with a variety of languages and dialects. For a current list of languages supported by Speakeasy, refer to the Wavelink Web site. To use Speakeasy with languages other than English, install the language (or international) build of the TE Client.
- **Integration with Avalanche.** When used with Wavelink Avalanche, Speakeasy client-side

application and configuration files can be easily deployed, updated, and archived wirelessly. Avalanche insures that when a mobile device starts, it has the latest application and configuration files.

Implementing Speakeasy

Speakeasy is software installed on a mobile device that can read text aloud to a device user or accept voice commands as text entry. It is designed to be used with the Wavelink Terminal Emulation (TE) Client. Use the following steps to set up Speakeasy:

1. Decide which emulation processes can be streamlined and made more efficient using either speech-to-text or text-to-speech.
2. Ensure you have the appropriate hardware, such as headphones and voice-enabled devices.
3. Obtain the desired licenses, speech-to-text packages, and text-to-speech packages from Wavelink.
4. Install the TE Client and the Speakeasy files on a development computer.
5. If you are using speech-to-text, create the list of words or phrases that the device users will need. Include the words and phrases in a grammar file. If you want to use voice training, specify in the grammar files which terms you want to train with.
6. Use the screen reformatter and TE scripting to handle the speech elements. This may include having the screen read aloud to the user (text-to-speech) or using voice commands to return text, execute actions, and change settings (speech-to-text).
7. Install the Speakeasy packages, grammar files, and scripts on the mobile devices and distribute the licenses to the devices.
8. If you implement voice training, users will be prompted to perform training when they start to use Speakeasy.

Using the Grammar File Manager

In order to use speech-to-text, you must configure the grammar files. Grammar files define what the Speakeasy engine will recognize. For example, the `digit.bnf` grammar file allows the engine to recognize when the speaker says any number zero through nine, "yes, please" or "no, thank you". Speakeasy comes with several default grammar files that can be edited to fit your environment.

The Grammar File Manager allows you to manage the following speech-to-text options:

- **Specify which grammar files are installed on your device**. Speakeasy comes with several grammar files that you can use. You can use them as provided, edit them, or create your own grammar files. Speakeasy allows you to deploy as many grammar files as you want to the device, and you can use up to ten grammar files simultaneously (during a script or while on a screen that has been modified using the screen reformatter).
- **Specify the prompts Speakeasy uses during calibration and user training**. You can create different prompts for each language you use with Speakeasy. The prompts are used when calibrating the microphone, when the user is creating a voice profile, and when voice profiles are being archived or retrieved from the Avalanche server.
- **Specify which words or phrases each user should train on before they begin using Speakeasy**. If there are words that the engine has trouble recognizing, you can have each user create a voice profile. When a user creates a voice profile, he speaks the words specified and Speakeasy stores the results and uses them to increase positive results.

After you make changes in the Grammar File Manager, you will need to deploy the changes to the mobile device. If you are using a Windows TE Client, you do not need to deploy the changes.

This section includes the following tasks for using the Grammar File Manager:

In previous versions, grammar files were considered either engine mode or attribute mode. Starting with version 2.0.5 of Speakeasy, all grammar files are considered engine mode grammar files.

Launching the Grammar File Manager

If you are using Wavelink Avalanche, the Grammar File Manager can be accessed through the Avalanche Console after you have the speech-to-text package added to a profile. If you are using ActiveSync to install or if you are using a Windows TE Client, then you can access the

Grammar File Manager through the Windows **Start** menu after you have installed Speakeasy on a PC.

Wavelink recommends you use the Windows TE Client for developing and testing grammar files, scripts, and reformatted screens.

To launch the Grammar File Manager from the Avalanche Web Console:

1 From the **Profiles** tab, select the software profile that has the speech-to-text base package.

2 Find the speech-to-text base package in the Software Packages panel and click **Configure**.

3 The *Configure Software Package* dialog box appears. Click **Next**.

4 Select **Grammar File Manager** from the list and click **Launch Config**.

To launch the Grammar File Manager for the Windows TE Client:

- Click **Start > Programs > Wavelink SpeechToText Support > Grammar File Manager**.

To launch the Grammar File Manager for an ActiveSync installation:

1 Create an ActiveSync connection to the device.

2 Click **Start > Programs > Wavelink SpeechToText ActiveSync Support > Install to Device**.

3 The *Windows Speech-to-Text Installer* appears. Click **Grammar File Manager**.

Importing and Exporting Grammar Files

Grammar files must be imported/exported using the Grammar File Manager in order to be modified or copied to a different package. When you export a grammar file from the Grammar File Manager, it is exported as a text file with a `.bnf` extension. The `.bnf` file can be copied or edited using a text editor such as Notepad. The file is then imported back into the Grammar File Manager so that it can be available for download to the device.

To import a .bnf file into the Grammar File Manager:

1 Launch the Grammar File Manager.

2 Click **Import**.

3 The *Import Grammar* dialog box appears. Navigate to the location where the grammar file is saved, select it, and click **Open**.

4 The *Select Languages* dialog box appears. Select all the languages from the list that the grammar will be used with and click **OK**.

5 The *Edit Item* dialog box appears. Type a description for the grammar file in the **Description** text box and click **OK**.

You can also edit the description for the grammar file after it has been added by selecting the file from the list in the Grammar File Manager and clicking **Edit**.

The grammar file is imported into the Grammar File Manager and you can select it to be downloaded to the device.

To export a grammar file from the Grammar File Manager:

1 Launch the Grammar File Manager.

2 Select the grammar file you want to export and click **Export**.

3 The *Save As* dialog box appears. Navigate to the location where you want to save the file and click **Save**.

The file is saved as a text file with a `.bnf` extension and can be modified with a text editor or imported using a different Grammar File Manager.

Editing a Grammar File

Grammar files define which words and phrases the Speakeasy engine will recognize. To edit a grammar file, launch the Grammar File Manager and use the Grammar Editor to make changes. Or, if you want to edit the grammar file using a text editor such as Notepad, export it from the Grammar File Manager, perform your edits, and import it back into the Grammar File Manager.

The Grammar File Manager comes with several grammars, but you can edit grammar files to contain the words you will use to perform tasks. For example, if you want to use a voice command to increase the speed of text-to-speech, add the term “faster” to a grammar file. Then use the screen reformatter to run a script for the speech-to-text result “faster” that increases the `tts_rate` setting. When the user says, “faster,” the text-to-speech speed increases.

To edit a grammar file:

1 Launch the Grammar File Manager.

2 Select the grammar file you want to edit and click **Edit**, or create a new grammar file by clicking **Create New Grammar**.

3 The Grammar Editor appears. Make changes to the file and then validate them by clicking the **Parse the grammar** button. If the compiler sees issues in the grammar file, the errors will appear in the pane at the bottom of the window.

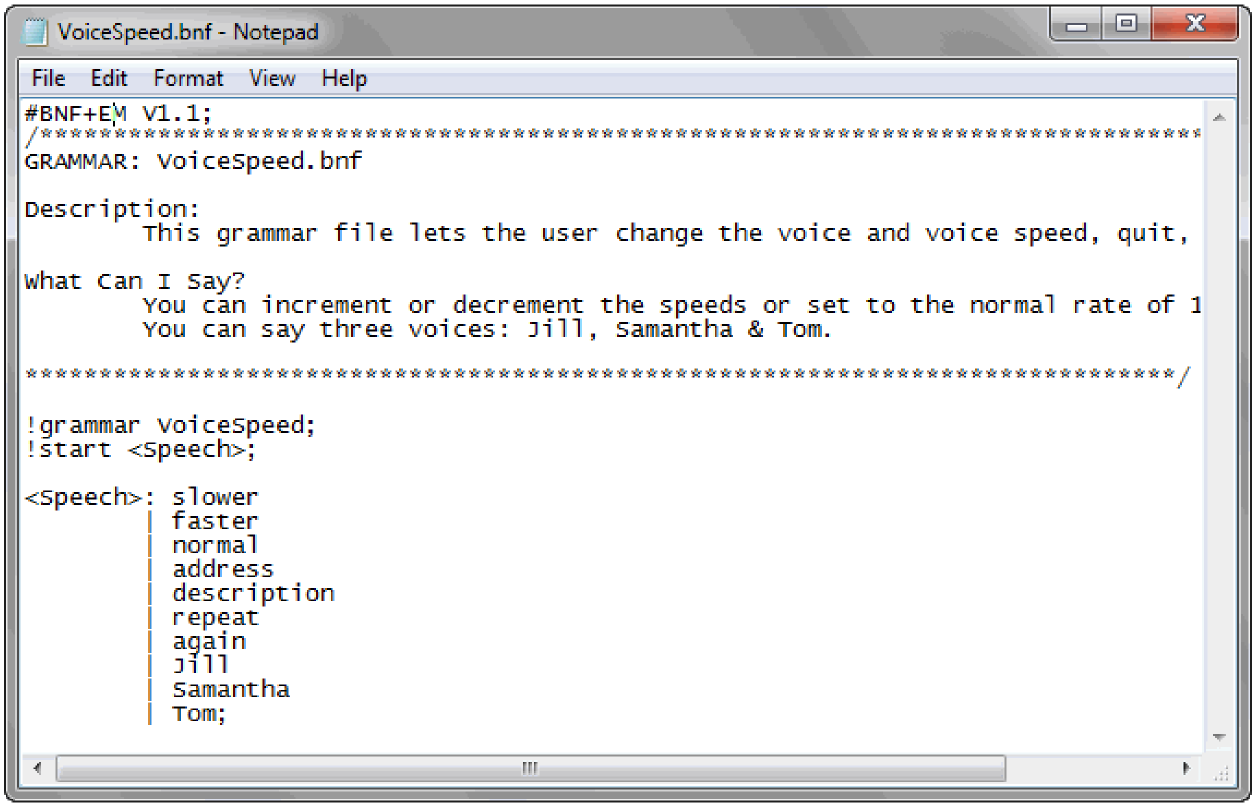
4 Click **Save**.

5 Close the Grammar Editor. The TE Client will update with the new grammar files the next time the device syncs.

The following sections provide information on how to edit grammar files:

Grammar File Structure

The following example is a basic grammar file (the grammar file `VoiceSpeed.bnf`) viewed in Notepad:



```
#BNF+EM V1.1;
/*****************************************************************************
GRAMMAR: VoiceSpeed.bnf

Description:
        This grammar file lets the user change the voice and voice speed, quit,

What Can I Say?
        You can increment or decrement the speeds or set to the normal rate of 1
        You can say three voices: Jill, Samantha & Tom.

*****************************************************************************/

!grammar VoiceSpeed;
!start <Speech>;

<Speech>: slower
        | faster
        | normal
        | address
        | description
        | repeat
        | again
        | Jill
        | Samantha
        | Tom;
```

Example of a Grammar File

Each grammar file must begin with a statement such as `#BNF+EM V1.1;` that identifies how Speakeasy should handle the grammar file. You do not need to modify this line.

To create comments — information ignored by the engine — you can begin a line with // or enclose text in `/*` and `*/` markers. You can edit the Description and the What Can I Say? sections to describe changes you make to the grammar file.

If you change the name of the grammar file, the command `!grammar` must use the new file name. For example, if the name of the file was changed to `VoicePick.bnf`, the command would read: `!grammar VoicePick;`

The `!start` command specifies which terms in the grammar will be available when the grammar is in use. In the above example, the engine will listen for all of the terms in the `<Speech>` rule. The words and phrases listed between `<Speech>:` and the last `;` are the terms the grammar will accept. Each word or phrase must be separated by | (an OR symbol). Terms may be listed all on the same line or aligned vertically for easier editing.

In some grammar files, the main rule <Speech> is divided into subsections to make it easier to manage the terms. In the example fragment below, the terms are divided so that the `!slot` command will only apply to some of the terms:

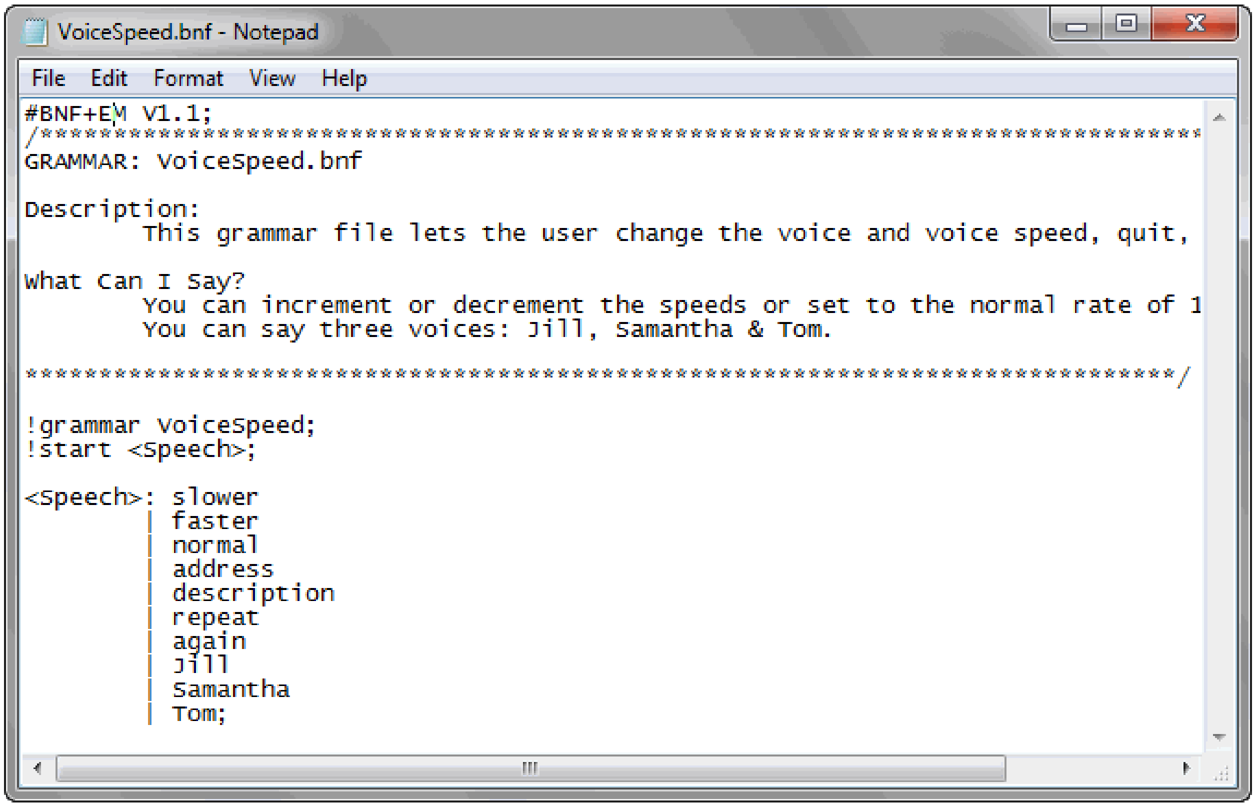
```
!start <Speech>;
!slot <YesNo>;

<Speech>: <YesNo> | <Maybe>;
<YesNo>: yes | no;
<Maybe>: maybe;
```

In this example, the <Speech> rule includes the <YesNo> and the <Maybe> rules, so the engine will listen for all the terms in both the <YesNo> rule and the <Maybe> rule. However, only the terms in the <YesNo> rule are included in the !slot command and used for voice training. (For more information about using the !slot command, see Enabling Voice Training for New Terms.)

Adding Terms to a Grammar File

A grammar file determines the words and phrases that the Speakeasy engine listens for. To add words or phrases to a grammar file, add it to an existing rule (such as `<Speech>`) and separate it from the other terms with a | symbol (also known as an OR symbol or a pipe). The following example is the grammar file `VoiceSpeed.bnf` viewed in Notepad:



```
#BNF+EM V1.1;
/**********************************************************************************
GRAMMAR: VoiceSpeed.bnf

Description:
        This grammar file lets the user change the voice and voice speed, quit,

What Can I Say?
        You can increment or decrement the speeds or set to the normal rate of 1
        You can say three voices: Jill, Samantha & Tom.

******************************************************************************/

!grammar VoiceSpeed;
!start <Speech>;

<Speech>: slower
        | faster
        | normal
        | address
        | description
        | repeat
        | again
        | Jill
        | Samantha
        | Tom;
```

Example of a Grammar File

In this example, the words and phrases listed between `<Speech>:` and the last `;` are the terms Speakeasy will recognize. Each word or phrase must be separated by an | symbol. Terms may be listed all on the same line or aligned vertically for easier editing.

To improve recognition, use longer and more distinct phrases. For example, rather than use the similar terms “faster” and “fastest,” replace them with the terms “go faster” and “warp speed”.

Once you have defined the words in a grammar file, use a script or the screen reformatter to define what happens when each word or phrase is recognized. Then deploy the grammar file and any other necessary files to the devices.

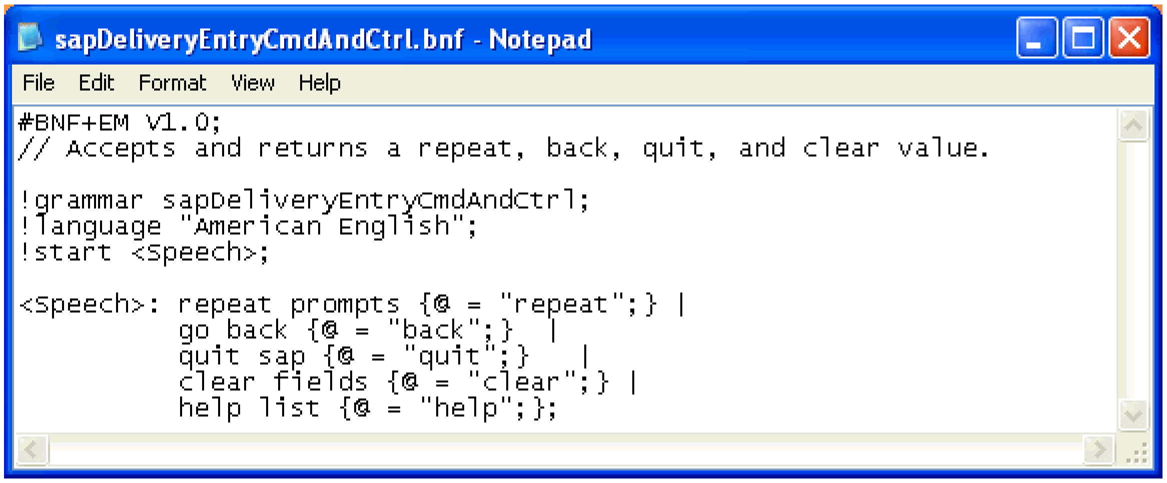
Using Alternate Return Values

You may want to have a grammar file return a value other than the exact phrase that the user says. This may be useful when you want to use phrases rather than words, or if you want different phrases to return the same result. Using phrases rather than single-syllable words improves recognition. An example of using an alternate return value could be that when the user says, "repeat prompts," the Speakeasy engine can act like it hears "repeat".

You can also use more than one word or phrase to return the same result. This may be especially helpful in a multilingual environment. For example, you could have an English grammar file that listens for "yes" and a Dutch grammar file that listens for "ja," but either grammar would return the result "yes." Then the script or reformatted screen would only have to be programmed for one result.

To use an alternate return value, list the word or phrase the engine should listen for and then `{@ = "result";}` where `result` is the alternate return value.

The following example is the grammar file `sapDeliveryEntryCmdAndCtrl.bnf` viewed in Notepad.


sapDeliveryEntryCmdAndCtrl.bnf - Notepad
File Edit Format View Help

```
#BNF+EM V1.0;
// Accepts and returns a repeat, back, quit, and clear value.

!grammar sapDeliveryEntryCmdAndCtrl;
!language "American English";
!start <Speech>;

<Speech>: repeat prompts {@ = "repeat";} |
          go back {@ = "back";}  |
          quit sap {@ = "quit";}    |
          clear fields {@ = "clear";} |
          help list {@ = "help";};
```

Example of a Grammar File

In this example, if the user said "go back," Speakeasy would return the text "back". However, if the user said "back," it would not match the terms the engine is listening for. Add a separate entry for "back" if you want the engine to listen for it, too.

Enabling Voice Training for New Terms

When you add words or phrases to a grammar file, you can specify if they should be available for voice training. The !slot command in a grammar file specifies which words and phrases should be available for voice training.

While you are adding terms to a grammar file, you separate the words you want to have available for user training from those you want to exclude from user training. Decide which terms you want to use for voice training, then separate them by creating a rule for each group. Then specify that both of the new rules belong to the main rule, Speech.

In the example below, the highlighted line shows that the rules YesNo and Cancel are included in the main rule, Speech. Then each term in the rules is defined on the last two lines.

```
!grammar YesNoCancel;
!start <Speech>;
!slot <YesNo>;

<Speech>: <YesNo> | <Cancel>;

<YesNo>: yes | no;

<Cancel>: cancel;
```

Notice that the !start command specifies the main rule, Speech. Since Speech includes YesNo and Cancel, this grammar file means the engine will listen for all the terms in both those rules ("Yes," "no," and "cancel").

After the terms have been grouped, use the !slot command on the line following the !start command (the third line in the example above). When you use the !slot command, specify the rule that you want to use for training. The example above will enable voice training for the terms in the rule YesNo.

The !slot command only works for rules that contain simple terms such as words, phrases, and alternate terms. It doesn't work for rules that contain other rules (such as Speech in the above example) or complex commands. You cannot use the !slot command for groups that contain !repeat or #.

Once you have put the terms in a rule and used the !slot command to specify the rule, use the Grammar File Manager to select the terms for training before the package is deployed to the device. See Specifying Voice Profile Training Options for information on selecting terms for user training.

Modifying User Prompts

Speech-to-text uses several prompts during calibration and voice profile training. These prompts default to English values, but you can modify the prompts for each language used with Speakeasy. If you are using a language other than English, there are no default values and you must configure the prompts in order to perform user training.

To modify the user prompts:

1 Launch the Grammar File Manager.

2 Click **User Prompts**.

3 The *User Prompts* dialog box appears. Select the language from the **Language** drop-down menu that the prompts will be available for, and modify the prompts as desired. If you are not using Avalanche to store and distribute voice profiles, you do not need to change the Training Data Archiving options.

4 Click **OK** to save your changes. The prompts will be used when you perform calibration or voice profile training.

Specifying Voice Profile Training Options

Speakeasy allows users to create personalized voice profiles, which can improve the positive results when you are using speech-to-text. When you create a voice profile, you speak a set of selected words to train the engine to recognize your voice and pronunciation. The words used for voice profile training come from the grammar files available on the device and are selected from the Grammar File Manager.

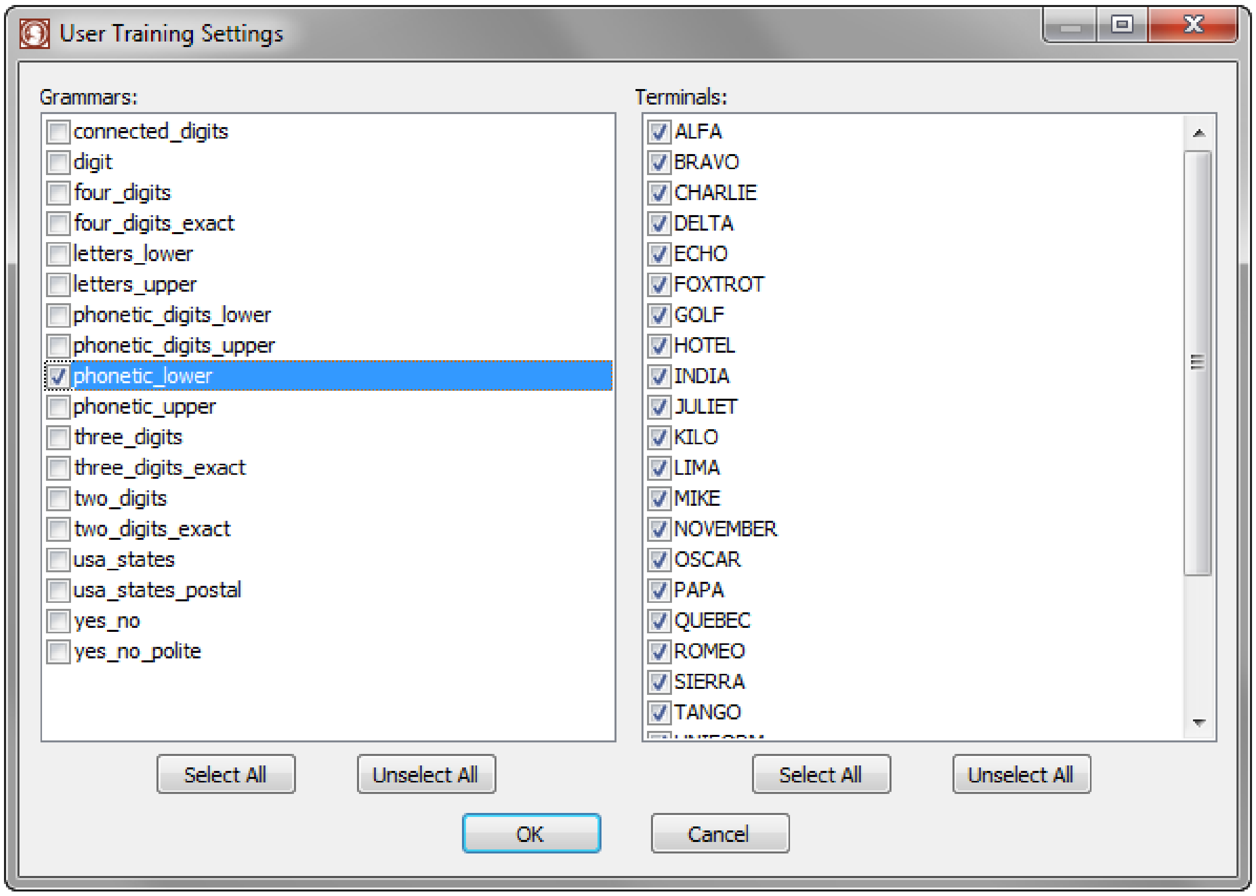
When you are selecting words and phrases to use for training, focus on terms that are short or that the engine may have trouble understanding. If you have two words that may sound similar or words that are only one syllable, you may want to include these in the training.

Before a word or phrase can be selected for voice training, you must define possible voice training words in the grammar file. For information on enabling voice training for words or phrases, see Enabling Voice Training for New Terms.

To specify the words used for voice profile training:

1 Launch the Grammar File Manager.

2 Click **Training Settings**.



Grammars:
- connected_digits
- digit
- four_digits
- four_digits_exact
- letters_lower
- letters_upper
- phonetic_digits_lower
- phonetic_digits_upper
- ✓ phonetic_lower
- phonetic_upper
- three_digits
- three_digits_exact
- two_digits
- two_digits_exact
- usa_states
- usa_states_postal
- yes_no
- yes_no_polite

Terminals:
- ✓ ALFA
- ✓ BRAVO
- ✓ CHARLIE
- ✓ DELTA
- ✓ ECHO
- ✓ FOXTROT
- ✓ GOLF
- ✓ HOTEL
- ✓ INDIA
- ✓ JULIET
- ✓ KILO
- ✓ LIMA
- ✓ MIKE
- ✓ NOVEMBER
- ✓ OSCAR
- ✓ PAPA
- ✓ QUEBEC
- ✓ ROMEO
- ✓ SIERRA
- ✓ TANGO

User Training Settings dialog box

3 The *User Training Settings* dialog box appears. From the **Grammar** list, select at least one grammar that has words or phrases that you want to use for training.

4 When you have selected a grammar, the words and phrases associated with that grammar appear in the **Terminals** list. Enable the checkboxes for the terms that you want to use for training.

If you know the word or phrase is in the grammar file but it doesn't show up in the Terminals list, you need to edit the grammar file to use the !slot command for that word or phrase.

5 When you have selected all the terms you want to use for training, click **OK**. The terms will be available for training when a voice profile is created or when a grammar is used that a user has not yet trained for.

For information on creating voice profiles, see Creating a Voice Profile.

Using Speakeasy with the Screen Reformatter

The TE Client screen reformatter allows you to modify the appearance of Telnet emulation screens. You can create a screen layout that includes items you want the mobile device user to see, and excludes items that should not be visible to the user. The screen reformatter also allows you to add Speakeasy actions or scripts to a screen.

This section lists tasks for the screen reformatter that are used with Speakeasy. For a full description of options available from the screen reformatter, see the *Terminal Emulation Client User Guide*. This section provides information about the following:

The Screen Reformatter does not work with Web emulation.

Adding Text-to-Speech in the Reformatter

You can add text to the modified screen that will be converted into speech and played back to the mobile device user. This can be text from the host screen or text added just for the modified screen. Multiple text-to-speech actions will be processed without a pause if they are adjacent. You can also use the screen reformatter to set text-to-speech settings.

To add text-to-speech from the screen reformatter:

1 Click **Edit > Add Text-to-Speech Text**.

-Or-

In the Initial Screen View, right-click and select **Add Text-to-Speech** from the context menu.

The *Text-to-Speech Text* dialog box appears.

2 Type your text in the **Text to be spoken** text box. This box can be left empty if you only want to change the text-to-speech settings.

3 Type the desired text-to-speech settings in the **Persistent Text-to-Speech Settings to use** and/or the **Temporary Text-to-Speech Settings to use** text boxes.

Each setting must start with `tts_` and use the format `setting=value`. Multiple settings can be specified and should be separated by commas. The complete list of settings is available in the *Terminal Emulation Scripting Reference Guide*.

4 Click **OK**.

The *Text-to-Speech Text* dialog box closes and your text appears in the Supporting Actions section of the Descriptive View. It doesn't show up on the emulation screen.

5 If desired, right-click on the Modified Screen View and select **Add Text**. Add the text to the modified screen.

To use existing text for text-to-speech:

1 In the screen reformatter, click and drag the mouse over the text you want to copy.

When you release the left mouse button, a context menu appears.

2 Select **Text-to-Speech Copy**.

The *Text-to-Speech Copy* dialog box appears.

3 If you want to change the current settings, type the desired text-to-speech settings in the **Persistent Text-to-Speech Settings to use** and/or the **Temporary Text-to-Speech Settings to use** text boxes.

Each setting must start with `tts_` and use the format `setting=value`. Multiple settings can be specified and should be separated by commas. The complete list of settings is available in the *Terminal Emulation Scripting Reference Guide*.

The Text-to-Speech action appears in the Supporting Actions section of the Descriptive View.

Adding Speech-to-Text in the Reformatter

You can add a speech-to-text action to the modified screen. This action converts the user's speech into text that will be processed according to the grammar file (or files) specified. It can print the text to the screen as keyboard data or perform an action associated with the command.

The **General** tab in the *Speech-to-Text* dialog box allows you to set the grammar files that Speakeasy should listen for when the user is on this screen. It also has the options for what Speakeasy should do with the result when it recognizes a word or phrase from any of the specified grammars. The actions include:

Print To Screen	Displays the result as text on the screen. **Keypress after result** sends a keypress after the result is printed on the screen.
Run Script	Instead of displaying the result on a screen, Speakeasy will start the specified script. Click **Edit** if you want to launch the script editor.
Parameters	Passes the specified parameters to the script when it begins.

Verify Result Asks the device user if the result is correct before printing it to the screen. If you enable this checkbox, the **Verify** tab appears in the *Speech-to-Text* dialog box. Use the options on the **Verify** tab to configure the verification question:

- Enter a question in the **Verification Question** text box. Use `#R#` to represent the speech-to-text result.
- Enter a grammar name in the **Verification Grammar to use** text box.
- Enter a response in the **Grammar Response if Right** text box.
- Enter a response in the **Text-to-Speech if Right** text box.
- Enter a response in the **Text-to-Speech if Wrong** text box.
- Enter a number (in seconds) in the **Timeout** text box. This is how long the screen reformatter will wait for verification that it repeated the correct result. If no verification is received, the speech result is discarded and no action is performed.

Delay Speech-to-Text to avoid feedback Ensures that the microphone ignores input while text-to-speech actions are happening. Enable this option if the device user is not using a headset microphone.

Speech-to-Text settings to use Changes a speech-to-text setting. Each setting must start with `stt_` and use the format `setting=value`. Multiple settings can be specified and should be separated by commas. If a value is not a number, then the Speech-to-Text engine will use the value closest to the value text description provided. The complete list of settings is available in the *Terminal Emulation Scripting Reference Guide*. Once a setting has been changed, that value will be used for future Speech-to-Text actions until it is changed again.

The **Local Actions** and **Global Actions** tabs in the *Speech-to-Text* dialog box allow you to assign actions to specific Speech-to-Text results. A Local Action's settings only apply to the current screen. Global Actions are shared among all the screens, so changing a Global Action for one Speech-to-Text support action will change the action for all the screens. Because the Local Actions take priority over Global Actions, you can override a Global Action by creating a Local Action for the same result value. For example, you could add a Global Action that lists available commands when the user says “help”. If there is a screen where not all the commands are available, use a Local Action to override the Global Action with a list of commands specific to that screen.

The following options are available for Local and Global Actions:

Speech-to-Text Result	The word or phrase you want to perform a specific action for. The speech-to-text result must be an exact match in order for the action to be performed. Speech-to-text results are case-sensitive.
Replace the result with this text	Replaces the speech-to-text result with the provided text. When this replacement is made, the **Keypress after result** value of the **General** tab is not used, so if you want to use an additional key press you should include it here. For instructions on determining the value of a key press, see "Performing a Keyboard Test" in the TE Client User Guide. Use `\` followed by the 4-digit hexadecimal number to specify a Telnet key press, and `\U` followed by the 4-digit hexadecimal number to specify a Unicode character. If you want to actually output a backslash character, use `\\` For example, to replace the result "euro" with "€" followed by a VT `Enter` key press, use this value: `\U20ac\000d`
Perform a key press	Replaces the speech-to-text result with a key press. You can use the name of the key (such as `F3` or `Enter`) or the hexadecimal number.
Start a script	Replaces the result with a script. Use an existing script or click **Edit Scripts** to launch the Script Editor. Assign initial values to the variables in the script by using a comma-delimited list inside of parentheses following the script name. For example, to run the script DoItNow and set the string variable **User** to **lucky** and the number variable **Position** to **17**, you would type this: `DoItNow (User="lucky",Position=17)`
Perform an IDA action	Replaces the result with the a standard action. Many of these actions are also available other ways, such as by writing a script or using a key press value.
Set a Speech-to-Text or Text-to-Speech setting	Replaces the result by changing the Speakeasy settings for the Client. Each setting must start with `stt_` or `tts_` and use the format `setting=value`. If a value is not a number, then the engine will use the value closest to the value text description provided. The complete list of settings is available in the *Terminal Emulation Scripting Reference Guide*. You can change multiple settings with a comma-delimited list (no spaces). The list must contain only Speech-to-Text settings, or only Text-to-Speech settings. You cannot mix the two types of settings. If you need to do that, then create and call a script instead of using this option.

Standard Beep	Replaces the result with a standard TE beep. If the beep has been modified in the emulation parameters, the modified beep will be used.
Error Beep	Replaces the result with a TE error beep. If the beep has been modified in the emulation parameters, the modified beep will be used.
Restart the supporting actions for this screen	Replaces the result with a restart of the supporting actions for the screen. Any text-to-speech actions and script actions will be repeated.
Cancel the Speech-to-Text	Interprets the result as a command to stop the Speakeasy listening engine while the user is on the current screen. The engine will begin listening again when the user changes screens.
Do Nothing	Ignores the result. This option is useful if a grammar is being used that can return results that don't apply to the current screen.

For each Speech-to-Text result received, the Local Actions are tested first. If no Local Actions match, then the Global Actions are tested. If no Global Actions match, the result will be treated as keyboard data and use the settings configured in the General tab.

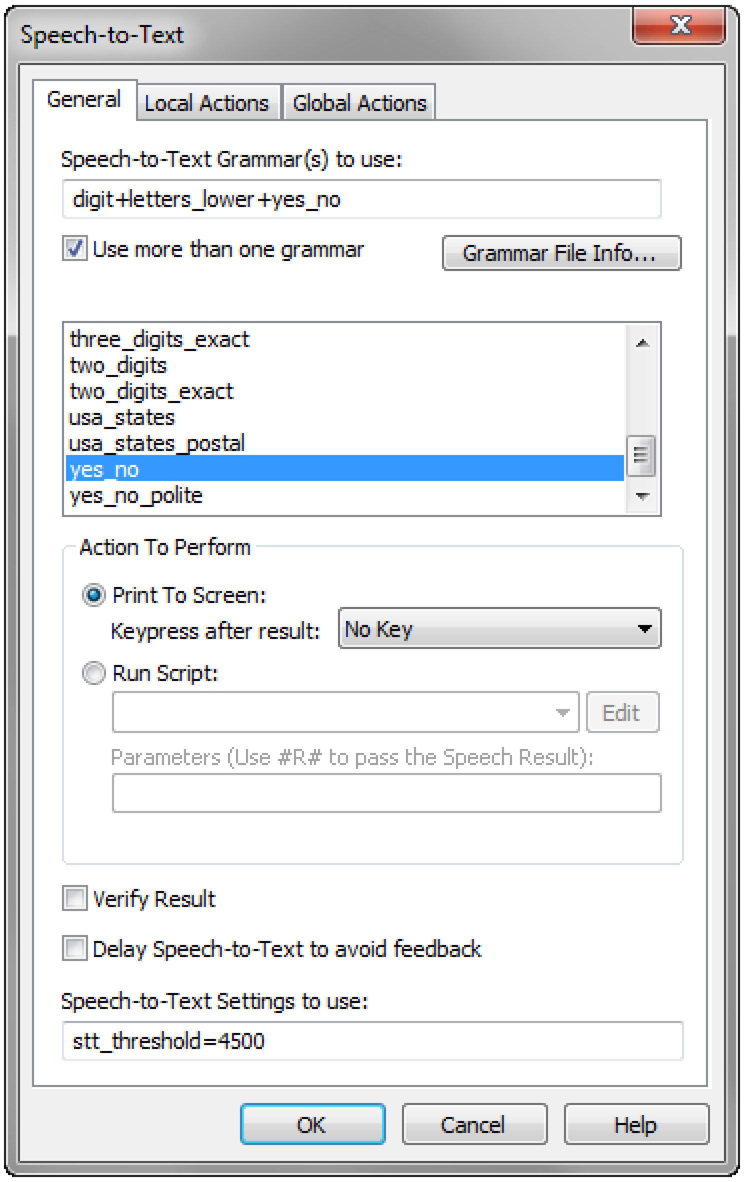
To add speech-to-text from the screen reformatter:

1 Click **Edit > Add Speech-to-Text**.

-Or-

In the Initial Screen View, right-click and select **Add Speech-to-Text** from the context menu.

The *Speech-to-Text* dialog box appears.



Speech-to-Text Grammar(s) to use: digit+letters_lower+yes_no

[✓] Use more than one grammar

three_digits_exact
two_digits
two_digits_exact
usa_states
usa_states_postal
yes_no
yes_no_polite

Print To Screen: [✓]
Keypress after result: No Key
Run Script:
Parameters (Use #R# to pass the Speech Result):

[ ] Verify Result

[ ] Delay Speech-to-Text to avoid feedback

Speech-to-Text Settings to use: stt_threshold=4500

Speech-to-Text options

2 Select the **Speech-to-Text Grammar to use** from the drop-down menu.

-Or-

Enable the **Use more than one grammar** option and choose the desired grammars from the list box.

The screen reformatter also supports dynamic grammar generation. In addition to (or instead of) using an existing grammar file, type a list of words or phrases separated by | (a pipe character) to generate a dynamic grammar. See Using the Grammar File Manager for more information on grammar files.

3 Select the action to perform with the result.

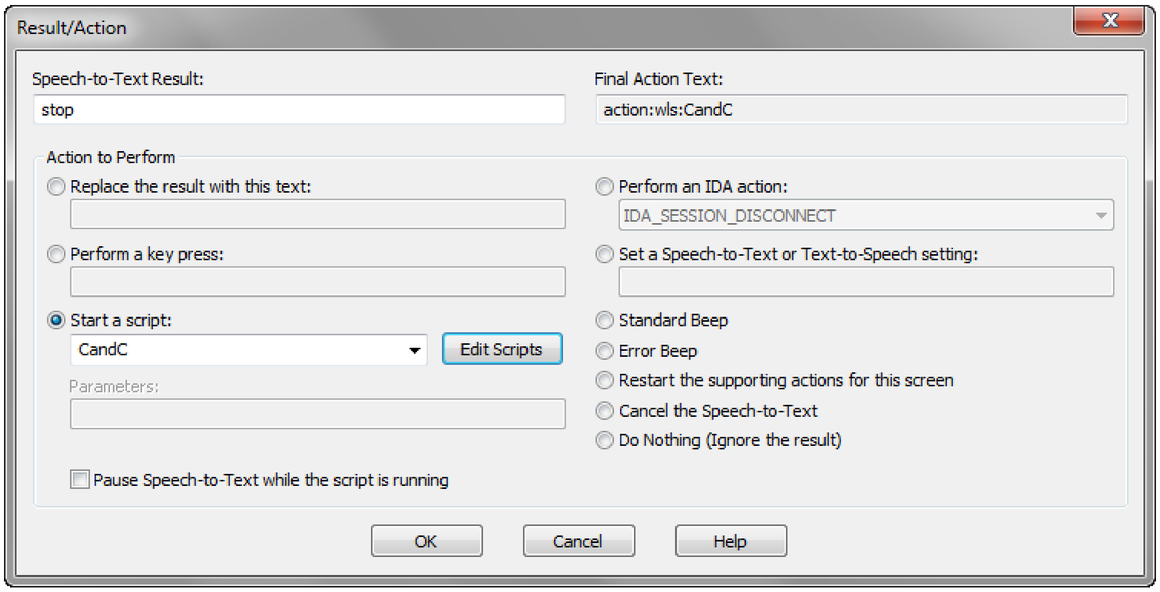
4 If you are not using a headset microphone, enable the **Delay Speech-to-Text to avoid feedback** checkbox. This will ensure that the microphone ignores input while text-to-speech actions are happening.

5 If you want to use a speech setting in the Speech-to-Text action, enter the setting in the **Speech-to-Text Settings to use** text box.

Each setting must start with `stt_` and use the format `setting=value`. Multiple settings can be specified and should be separated by commas. If a value is not a number, then the Speech-to-Text engine will use the value closest to the value text description provided. The complete list of settings is available in the *Terminal Emulation Scripting Reference Guide*. Once a setting has been changed, that value will be used for future Speech-to-Text actions until it is changed again.

6 If you want to configure local or global actions for the screen, select the appropriate tab and click **Add**.

The *Result/Action* dialog box appears.


Result/Action

Speech-to-Text Result: stop

Final Action Text: action:wls:CandC

Action to Perform

- Replace the result with this text:
- Perform a key press:
- [✓ Start a script:] CandC
- Parameters:
- Pause Speech-to-Text while the script is running
- Perform an IDA action: IDA_SESSION_DISCONNECT
- Set a Speech-to-Text or Text-to-Speech setting:
- Standard Beep
- Error Beep
- Restart the supporting actions for this screen
- Cancel the Speech-to-Text
- Do Nothing (Ignore the result)

Result/Action for Speech-to-Text

7 Configure the options to perform the action for the result. The Final Action Text box will display the text that represents the action and will be displayed on the **Local Actions** or **Global Actions** tab. Click **OK** to save the changes and return to the *Speech-to-Text* dialog box.

8 When you have configured speech-to-text for the modified screen, click **OK**.

The *Speech-to-Text* dialog box closes and the speech-to-text action is added to the modified screen.

If you need to modify the speech-to-text action, right-click the action in the Descriptive View and select **Edit Speech-to-Text**.

Adding Scripts to Reformatted Screens

Use the screen reformatter to launch Speakeasy scripts when the modified screen is first displayed. The script can provide additional functionality to the modified screen.

To add a script from the screen reformatter:

1 Click **Edit > Add Scripting Support**.

-Or-

In the Initial Screen View, right-click and select **Add Scripting Support** from the context menu.

The *Scripting Support* dialog box appears.

2 Select the desired script from the **Script to launch** drop-down menu. If you want to pass specific parameters to the script, type them in the **Parameters** text box separated by commas.

The **Script to launch** drop-down menu only displays scripts that have been saved in the Script Editor. For more information about Terminal Emulation scripting, see the *Wavelink Terminal Emulation Scripting Reference Guide.*

3 Select how the screen reformatter will handle the script:

- If you want to ensure that the script does not run multiple times, enable the **Don't launch the script if it is already running** checkbox.
- If you want the script to abort when the modified screen is no longer in use, enable the **Stop the script when reformatting changes** option.
- If you want the screen reformatter to wait until the script has completed before it proceeds to the next action for the screen, enable the **Wait for the script to finish before performing the next supporting action**.

4 Click **OK**.

The *Scripting Support* dialog box closes and the script is added to the Supported Actions listed in the Descriptive View.

Ordering Screen Actions

When you have multiple actions on the modified screen, you can determine the order in which the actions occur. The actions are listed in the Descriptive View in the Supported Actions section. Supported screen actions are Speakeasy and scripting actions. You should list the text-to-speech actions before any speech-to-text actions.

To arrange actions in the desired order:

- In the Descriptive View, right-click the item that you want to modify and select one of the following options:
  - **Make First Action** to designate the current action as the first action to be performed.
  - **Make Next Action** to designate the current action as the next action in the list.
  - **Make Previous Action** to designate the current action as the previous action in the list.
  - **Make Last Action** to designate the current action as the last action to be performed.
  - **Delete Action** to remove the current action.

Deploying Reformatted Screens

Once you have finished modifying your screens, click **Save** to save your screens and exit the screen reformatter. The TE Client will update with the new screens the next time the device syncs.

Using Reformatter Logging

The Telnet log configuration file, `wllogcfg.txt`, is used to enable Telnet application, Screen Reformatter, and Telnet network logging. When this file is used, the TE Client creates another file containing logging information for that session. The parameters used to enable screen reformatter and network logging are `screenreformatter` and `networklogging`.

Network logging is provided by the network traffic log file feature. This feature allows the network data to be viewable in the same log file as Telnet and Screen Reformatter data.

To use Reformatter logging, you must be using TE v7.3.190 or newer.

The `screenreformatter` parameter can be set to one of the following options:

- **0:** Disables the logging feature completely.
- **1:** Shows the criteria used to remap the screen.

- **2:** Shows why the screen remap stopped and the first item that caused it to fail.
- **3:** Shows the complete discovery process for identifying the screen remap failure, including attempted screens and the final screen used.

The `networklogging` parameter can be set to `0` or `1`, with `1` enabling network logging. The `screenreformatter` parameter requires that the Network Traffic Log File emulation parameter be enabled from the Client.

To enable these files in the TE Client:

1. Navigate to `[TE Client installation location]\` and create a text file.
2. In the file, enter the following with the values for `screenreformatter` and `networklogging` parameters set as desired:

   ```
   file=C:\temp\newTEloginfo.txt

   level=0

   max=1000000

   screenreformatter=1

   networklogging=1
   ```

3. Save the file with the name `wllogcfg.txt` and close it.

The `wllogcfg.txt` file must reside within the same folder as the `Telnet.exe` or `TelnetCE.exe` files.

4. To enable network logging on the Telnet Client, go to **Term > Configure > Emulation Parameters**.
5. Navigate to **Emulation > Logging > Network Traffic Log File**.
6. Enter the full file path to the `newTEloginfo.txt` file, including the file name.
7. Click **OK** and then save and restart the TE Client.

   The `newTEloginfo.txt` file is created and any Reformatter and networking logging collected is populated into the file.

When your logging diagnostics are no longer needed, Wavelink recommends moving the `newTEloginfo.txt` and `wllogcfg.txt` files to another folder on a PC for future use. Remove the `newTEloginfo.txt` and `wllogcfg.txt` from the mobile device. Each time you run logging, a new `newTEloginfo.txt` file is created for that logging instance and will overwrite any previous data.

Speakeasy and Scripting

Use the Terminal Emulation Script Editor to create and execute scripts that automate Speakeasy processes, such as selecting the right voice profile or changing the speed of text-to-speech. A script can be started using the Screen Reformatter, a key combination or scan input, calling it from another script or a web page, or from the TE Client by clicking **Options > Scripting > Execute Script**. You also have the option of launching a script when the session connects or when the screen updates.

When using text-to-speech and scripting, you must convert an integer variable to a string variable in order for text-to-speech to read it.

This section provides overview information for the following topics:

For examples of Speakeasy scripts, see Sample Speakeasy Scripts.

Creating Scripts

The following steps provide an overview of how you manually create a Speakeasy script. For more detailed information about these steps or scripting actions, refer to *Wavelink Terminal Emulation Scripting Reference Guide.*

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 how you want the script to perform.

For actions specific to Speakeasy, refer to Speakeasy Settings.

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.

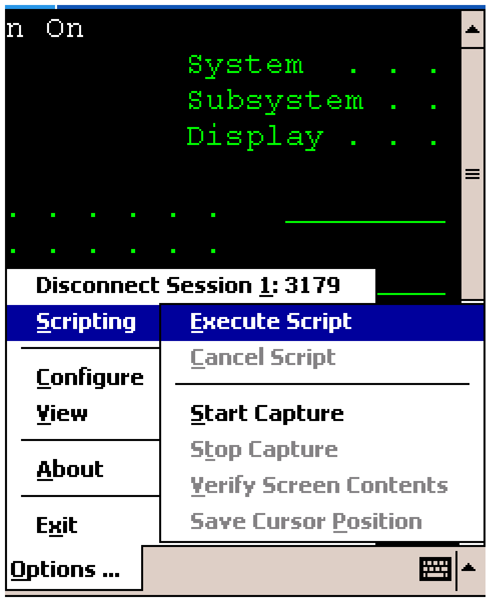
Activating Scripts

When a script is created, it has an activation method assigned that specifies how it is activated. Scripts are only available when a session is connected. You can have more than one script running at a time. This section provides information about activating scripts using each of the configurable activation methods.

If you plan to call a script from another script, a web page, or from the screen reformatter, you do not need to select an activation method for the script in the Script Editor.

Select From Menu

Activate the script from the **Options** or **Term** menu of the TE Client. From the **Options** menu of the TE Client, select **Scripting > Execute Script**. If more than one script is available for the current host profile, select which script you want to use from the list.



Executing Scripts from the Menu

From Another Script

To use a script to call another script, use the following format:

```
Call ("Name of the script",argument1,argument2,arugment3,argument4)
```

The arguments are optional and you cannot use more than four of them. They can be variables, strings, Booleans, or numbers. If the argument is a string, it must have quotation marks around it.

On Key Combination

Activate the script by pressing the specified key combination. The script will run if it is currently possible for a script to run.

When Session Connects

The script activates when a session connects using the specified host profile.

On Barcode, MSR, or RFID Scan

The script activates with each barcode, MSR, or RFID scan.

On Screen Update

The script activates (if activation is allowed) every time the text on the emulation screen changes. This includes updates from the host or when the user presses a key and the key value appears on the screen.

From the Screen Reformatter

A Wavelink script can be launched for a specific screen using the screen reformatter. When a script is added for a screen in the reformatter, it is considered a screen action. When the correct screen appears on the device, screen actions (including speech-to-text and text-to-speech actions) are performed in the order they appear in the Descriptive View. For information on adding a script to a screen using the screen reformatter, see Adding Scripts to Reformatted Screens.

From Web Pages

Activate the script for the Industrial Browser from a web page using the `wls` type, followed by the script name. If you plan to launch a script from a web page, do not select a script activation method when you create the script.

Executing Scripts from Web Pages Example 1

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

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

Executing 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>
```

Colliding Speech Engine

Colliding speech instances occur when two speech-to-text instances are running at the same time. When two speech-to-text instances collide, this can prevent users from progressing to another screen or performing verbal commands. In the instance of such collisions, Speakeasy alerts you of the conflict and cancels the script with the lowest priority.

Three general conditions exist to cause a colliding speech scenario:

1. A script with a speech-to-text instance is launched when a speech-to-text instance launched by a script is already running.

2. The Reformatter launches a speech-to-text supporting action when a speech-to-text instance launched by a script is already running.

3. A script launches a speech-to-text instance while a speech-to-text instance launched by a Reformatter supporting action is already running.

Priorities are determined by a script's order or the presence of Screen Reformatter actions. The logic used to determine priorities is described below:

- **Script followed by another script.** When a script is followed by another script that contains a speech-to-text supporting action, the second speech-to-text instance takes precedent and forces the first speech-to-text instance to stop.

- **Screen Reformatter speech-to-text supporting action followed by a script.** When a Screen Reformatter speech-to-text supporting action is followed by a script, the Reformatter action takes precedent and prevents the script from starting.

- **Script followed by a Screen Reformatter speech-to-text supporting action.** When a script is followed by a Screen Reformatter speech-to-text supporting Reformatter action, the speech-to-text supporting action takes precedent and forces the script to stop.

When one of these scenarios occurs, an alert appears, indicating the name of the script that was canceled.

Installation and Licensing

Speakeasy consists of multiple packages (in addition to the Terminal Emulation Client) that must be deployed to the mobile device. You can install Speakeasy packages on the mobile device using Wavelink Avalanche or Microsoft ActiveSync. You can also install Speakeasy packages for the Wavelink TE Client for Windows.

To improve speech recognition and positive results, Speakeasy speech-to-text uses grammar files to define expected input. If you are using speech-to-text, you should use the Grammar File Manager to modify and select the grammar files you plan to use before you install speech-to-text on the device. The Grammar File Manager is available whether you are installing via Avalanche or ActiveSync.

This section provides the following information on installing Speakeasy:

Wavelink supports some third-party installation applications. For more information about supported installation options for your device, please see the Wavelink Web site. If you choose to use a third-party application to configure and install Speakeasy, please see the documentation for that application.

Installation Requirements

This section lists the hardware, software, and memory requirements that Speakeasy requires for best performance:

- Mobile device with headset microphone with a signal-to-noise ratio (SNR) better than 20 dBA
- 128 MB RAM; or 64 MB RAM with an SD card; or 64 MB RAM with 128 MB Flash Memory
- Wavelink Terminal Emulation Client version 7.0 or newer (for best performance, Wavelink recommends TE client version 7.3.59 or later)

If you are using Speakeasy with a desktop TE Client, Speakeasy requires:

- Windows XP or later
- Minimum of 2 GB disk space

If you want to be able to save your voice profiles and distribute them using Wavelink Avalanche, you will also need:

- Avalanche Mobility Center or Avalanche Site Edition. You must use a mobile device server that is version 5.2 or newer.
- Wavelink Avalanche Enabler version 5.0-10 or newer.

Installing Speakeasy Using Avalanche

When you install Speakeasy on a device using Avalanche, each device that will be using Speakeasy (either speech-to-text or text-to-speech) must receive the Speech Registration package. If you are planning to use speech-to-text, you will also need the following packages:

- Speech-to-text Base package.
- Speech-to-text Language packages. Choose the package(s) specific to the language(s) you will be using.

If you are planning to use text-to-speech, you will need the following packages:

- Text-to-speech Base package.
- Text-to-speech Vocalizer packages. Choose the package(s) specific to the language(s) you will be using.

You should have the Terminal Emulation Client installed before installing Speakeasy on the device.

The following instructions are for installing Speakeasy using the Avalanche Web Console. For instructions on installing using the Java Console, see the *Avalanche Java Console User Guide*.

To install Speakeasy using Avalanche:

1. Download the applicable packages from the Wavelink Web site.
2. If desired, use Avalanche to create a new software profile for the Speakeasy packages.
3. Add the Speakeasy packages to the software profile and navigate to the Software Profile Details page.
4. If you want to specify where on the device you want the Speakeasy packages installed, configure the registration package. You should install the packages in a location with sufficient memory that is cold-boot persistent.
5. If you are using speech-to-text, configure the speech-to-text base package. Use the grammar file manager to configure the grammars the device will need. The grammars define which words the speech-to-text engine will recognize. For more information on grammar files, see Using the Grammar File Manager.
6. Apply the software profile to the locations where you want the packages to be distributed and enable the profile.
7. Perform a synchronization.

8 If desired, from the Enabler interface on the mobile device, click **File > Connect** to immediately connect to the mobile device server, download the packages, and install them.

For more information on performing tasks from the Avalanche Console, refer to the Avalanche User Guide.

Installing Speakeasy Using ActiveSync

Speakeasy can be installed on a mobile device using Microsoft ActiveSync or the Windows Mobile Device Center. There are separate installation files for speech-to-text and text-to-speech. Use the package labeled for ActiveSync if you are using the Windows Mobile Device Center.

If you are planning to use speech-to-text, you will need the following package:

- Speech-to-text ActiveSync installation package.

If you are planning to use text-to-speech, you will need the following package:

- Text-to-speech Vocalizer packages. Choose the package or packages specific to the languages you will be using.

You should have the Terminal Emulation Client installed before installing Speakeasy on the device.

To install speech-to-text using ActiveSync:

1 Download the applicable packages from the Wavelink Web site.

2 Establish a connection with the device.

3 Launch the ActiveSync installation speech-to-text package by double-clicking on the file.

4 The Setup Wizard appears. Click **Next**.

5 The License Agreement appears. Accept the agreement by clicking **I Agree**.

6 The Choose Installation Location screen appears. Click **Browse** to select a different installation location, or click **Install** to use the default location.

The Speech-to-text files are installed locally.

7 When the **Completing the Setup Wizard** screen appears, ensure that the **Run Wavelink Speech-to-Text ActiveSync Support** checkbox is enabled and click **Finish**. The wizard will close and the Speech-to-Text Installer will launch.

8 From the **Languages** list, select the languages you will be using.

9 To choose the grammars that will be installed on the device, click **Grammar File Manager**. For more information on managing grammar files, see Using the Grammar File Manager.

10 Click **Install** to install speech-to-text on the device. When the files are finished installing, the *Installation Successful* screen appears. Click **OK**.

To install text-to-speech using ActiveSync:

1 Download the applicable packages from the Wavelink Web site.

2 Establish a connection with the device.

3 Launch the ActiveSync installation text-to-speech package by double-clicking on the file.

4 The Setup Wizard appears. Click **Next**.

5 The License Agreement appears. Accept the agreement by clicking **I Agree**.

6 The Choose Installation Location screen appears. Click **Browse** to select a different installation location, or click **Install** to use the default location.

The text-to-speech files are installed locally.

7 When the **Completing the Setup Wizard** screen appears, ensure that the **Run Wavelink Text-to-Speech ActiveSync Support** checkbox is enabled and click **Finish**. The wizard will close and the Text-to-Speech Vocalizer Installer will launch.

8 From the **Languages** list, select the languages you will be using.

9 Click **Install** to install text-to-speech on the device. When the files are finished installing, the *Installation Successful* screen appears. Click **OK**.

Installing Speakeasy on a PC

Speakeasy can be installed on a Windows PC that has the Windows TE Client installed. Depending on your organization's needs, you may choose to install only speech-to-text, or only text-to-speech. Download the files from the Wavelink Web site and copy them to the computer where the TE Client is installed.

To install speech-to-text:

1 Double-click the Speech-to-Text executable.

The *Wavelink SpeechToText Support Setup Wizard* appears.

2 Follow the instructions in the Speech-to-Text Support Setup Wizard to complete the installation.

3 When the installation is complete, click **Finish** to exit the Wizard.

To install text-to-speech:

1 Double-click the Text-to-Speech Vocalizer executable.

   The *Wavelink TextToSpeech Vocalizer Support Setup Wizard* appears.

2 Follow the instructions in the Text-to-Speech Vocalizer Support Setup Wizard to complete the installation.

3 When the installation is complete, click **Finish** to exit the Wizard.

Licensing

Speakeasy requires a separate license in addition to the standard Terminal Emulation licenses. You can use Speakeasy without a license, but you will be limited to the demo version. Speakeasy licenses can be distributed using the same method as other TE licenses. For information on licensing methods or distributing licenses, see the *Terminal Emulation Client User Guide*. To obtain licenses, please contact Wavelink Customer Service.

Uninstalling Speakeasy

Speakeasy can be uninstalled from the PC you used to install or from the mobile device. Uninstalling Speakeasy does not uninstall the TE Client.

Uninstalling Speakeasy from a Mobile Device

Speakeasy is installed on a mobile device using Avalanche or ActiveSync. To uninstall Speakeasy from a mobile device, use the same application that you used to install.

The following instructions are for uninstalling Speakeasy from all devices. You can also uninstall Speakeasy from a specific device or region by changing how you want the Speakeasy software profile applied.

To uninstall Speakeasy using Avalanche:

1 From the Avalanche Console, navigate to the profile containing the Speakeasy packages.

2 Select the Speakeasy packages and click **Delete**.

3 Perform a synchronization. When a package is deleted from the Avalanche Console, the package on the device is considered orphaned. You can delete orphaned packages from devices using a mobile device profile or by using the **Update Now** option on the Avalanche Console.

To uninstall Speakeasy using ActiveSync:

1. Create an ActiveSync connection to the device.
2. From the PC, click **Start > Programs > Wavelink SpeechToText ActiveSync Support > Install to Device** to uninstall the speech-to-text files.

   -Or-

   From the PC, click **Start > Programs > Wavelink TextToSpeech ActiveSync Support > Install to Device** to uninstall the text-to-speech files.
3. The Wavelink Installer appears. In the **Installation Location** drop-down menu, select the location where the files were originally installed.
4. Ensure that all the languages are deselected and click **Install**.
5. The files are removed from the device.

Uninstalling Speakeasy from a PC

Speakeasy can be used for a Windows TE Client or deployed from a PC to a mobile device via ActiveSync or Avalanche. If you used Avalanche to install, you do not need to uninstall the ActiveSync support files from the PC.

To uninstall Speakeasy for a Windows TE Client from a PC:

- To uninstall speech-to-text support, click **Start > Programs > Wavelink SpeechToText Support > Uninstall SpeechToText Support**.

  -Or-

- To uninstall text-to-speech support, click **Start > Programs > Wavelink TextToSpeech Vocalizer Support > Uninstall TextToSpeech Vocalizer Support.**

To uninstall Speakeasy ActiveSync Support from a PC:

- To uninstall speech-to-text ActiveSync support, click **Start > Programs > Wavelink SpeechToText ActiveSync Support > Uninstall SpeechToText ActiveSync Support**.

  -Or-

- To uninstall text-to-speech ActiveSync support, click **Start > Programs > Wavelink TextToSpeech ActiveSync Support > Uninstall TextToSpeech ActiveSync Support**.

Using Speakeasy

Once Speakeasy is installed on the device, the TE Client has tools to optimize Speakeasy. You can calibrate the microphone or text-to-speech options, create a voice profile, or enable the Speakeasy listening indicator to show when speech-to-text is listening.

When you create voice profiles, each person can create his own profile in order to optimize results. If you are using Speakeasy in conjunction with Avalanche, voice profiles can be stored on the mobile device server and distributed to devices as needed. Otherwise, each user will need to create a voice profile on each device he plans to use.

Calibrating the Microphone

On some mobile devices, you can calibrate the microphone to optimize the speech detection in your current environment. The microphone settings that you select will become the default values for future speech-to-text processing on the mobile device. It is recommended that you calibrate the microphone before initial use.

You can perform a quick calibration that automatically sets values according to your speaking voice and the background noise, or you can perform a full calibration, manually setting the values using slider bars. Before you begin calibration, ensure that your headset is connected and working properly.

When the speech-to-text engine hears input, it tries to match it to the expected phrases in the available grammar files. The degree to which the spoken phrase and an expected response match is called the confidence value. A higher confidence value indicates a better match. If the confidence value is too low, the input is rejected.

You do not have the option of a quick calibration unless Text-to-Speech is installed.

To perform a quick calibration:

1 From the Client **Options** menu, select **Configure > Microphone**.

2 The *Options* dialog box appears. Click **Quick Calibration**.

3 The *Language* dialog box appears. Select the language you want to use from the drop-down list and click **OK**.

The voice prompts will guide you through a prompt for speaking and a prompt for silence. During the speaking prompt, the top 10 energy levels are collected and the average used. The default test prompt is "Testing 1-2-3-4-5", as this longer prompt collects more energy levels for evaluation. When the calibration is complete, the new settings are applied and become the defaults for the device. For information on modifying the calibration prompts using the Grammar File Manager, see Modifying User Prompts.

To perform a full calibration:

1 From the Client **Options** menu, select **Configure > Microphone**.

2 The *Test Settings* dialog box appears. Click **Full Calibration**.

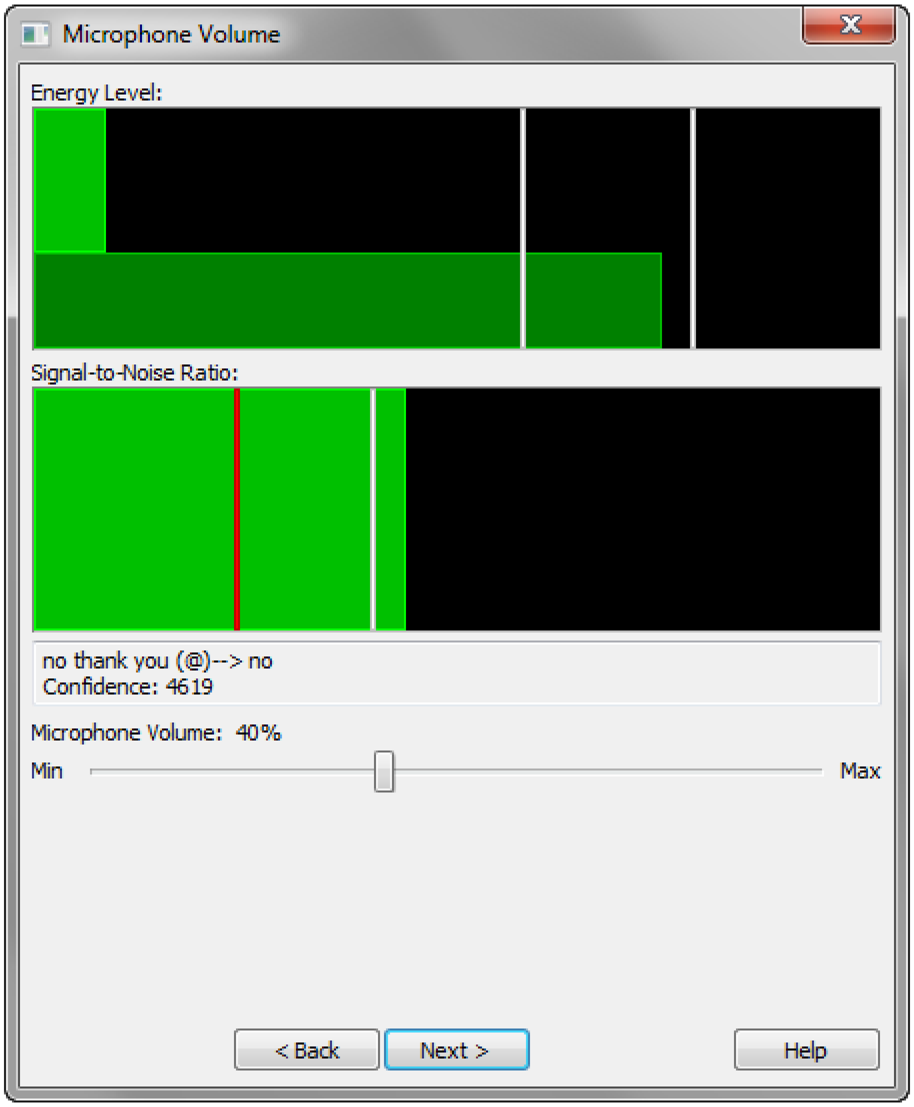
3 From the **Language** drop-down menu, select the language you will be using for speech-to-text conversion.

4 From the **Grammar** drop-down menu, select the grammar you will be using for speech-to-text conversion.

5 If you have a session currently connected, you can use the speech-to-text settings for that session instead of the default settings. Select the session from the **Session** drop-down menu.

6 Click **Next**.

The *Microphone Volume* dialog box appears.



no thank you (@)--> no
Confidence: 4619

Microphone Volume: 40%

Microphone Volume Dialog Box

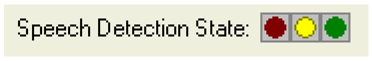
7 Use the **Microphone Volume** slider to adjust the Energy Level.

The Energy Level indicates how much the microphone input is currently changing. The top bright green bar is the current energy level, and it should fall between the two white bars when the user is speaking. The darker bar underneath is the maximum level reached within the past 5 seconds. The Signal-to-Noise Ratio bar shows the quality of the microphone and device audio processing. After you start talking, this bar should be to the right of the white line

When you speak one of the words or phrases included in the grammar selected in step 4, the phrase appears in the box underneath the Signal-to-Noise Ratio. This box also displays the confidence value.

8 Click **Next**.

The *Other Settings* dialog box appears. The **Speech Detection State** indicates whether the speech engine detects the user's speech. When the user is not speaking, the state should be red. When the speech engine detects something that may be speech, the state turns yellow. When the speech engine is certain it is detecting speech, the state turns green.



Speech Detection State

9 Use the **Absolute Threshold** and **Sensitivity** sliders to adjust the speech detection settings.

The **Absolute Threshold** value indicates the minimum amount of sound required to indicate when speech begins. Adjust this value so that the user's speech causes the state to turn yellow immediately, but any background noise causes the state to remain red.

The **Sensitivity** value determines when the speech engine begins detecting a user's speech. A higher sensitivity value means the speech engine will react more easily; a lower value means the speech engine will pick up less background noise.

10 Click **Next**.

The *Record/Playback* dialog box appears. If you want to check the confidence values for a recording against a grammar file, you can use the options on this dialog box to record and play back microphone input as a `.wav` file. This option is helpful if you have words that the engine has a hard time recognizing. Record someone saying the words in a `.wav` file, and check the confidence levels using the playback. Modify the grammar files and then check back to see if your changes have increased recognition (higher confidence values).

11 Click **Finish**.

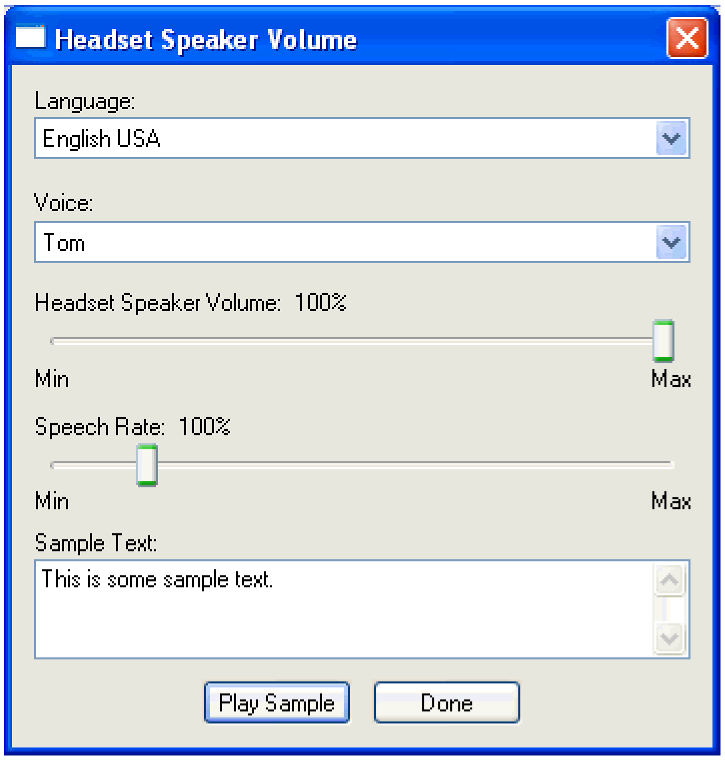
The microphone calibration wizard disappears and the new microphone settings are applied. These settings will be the defaults for the device.

Configuring the Text-to-Speech Options

On some mobile devices, you can configure the speaker volume and speed settings. The speaker settings that you select will become the default values for future text-to-speech processing on the mobile device.

To configure text-to-speech options:

1 From the TE Client, click **Options > Configure > Speaker Volume**.



Speaker Settings

The *Headset Speaker Volume* dialog box appears.

2 From the **Language** and **Voice** drop-down menus, select the desired language and voice. The options that appear in the **Voice** drop-down menu are dependent on the language selected.

3 Use the sliders to set the volume and how fast the text-to-speech is pronounced.

4 If desired, type sample text in the **Sample Text** box and click **Play Sample** to check your settings.

5 When you have configured the settings, click **Done**.

The speaker settings wizard disappears and the new speech settings are applied.

Creating a Voice Profile

A voice profile provides the Speakeasy engine with speech samples in order to get more positive results during speech-to-text. You can create a voice profile from the TE Client after Speakeasy is installed.

Before you create a voice profile, you should have the grammar files configured. When you configure grammar files, you select the specific words and phrases that users will train on to create voice profiles. For more information on selecting the terms for voice profiles, see Specifying Voice Profile Training Options.

Once you have created a voice profile, you can use a script to implement the profile. For an example script that allows you to enter the username and use a voice profile, see Using a Voice Profile.

If you are using an Avalanche Enabler and mobile device server, voice profiles are archived on the mobile device server. If you enter the same username on a different device, the TE Client will automatically retrieve the voice profile from the mobile device server and use it.

To create a voice profile:

1 Ensure that your headset is connected.

2 From the TE Client, click **Options** > **Configure** > **Microphone**.

3 The *Options* dialog box appears. Click **User Training**.

4 The *Language* dialog box appears. Select the language you want to use from the drop-down menu and click **OK**.

5 The *User Name* dialog box appears. Type the user name in the text box and click **OK**.

6 The User Training dialog box appears. Speakeasy vocally prompts you to say the words and phrases selected for user training when the grammar files were configured. You are prompted to say each word or phrase twice. After all the selected terms have been recorded, the profile is saved.

If you only want to train for some of the available terms, stop the training and select the checkboxes next to the terms you want to train for. Then click **Start**.

If you create a voice profile and later add a grammar file that requires additional training, you must use this process again in order to complete the training.

Using the Speakeasy Listening Indicator

The Speakeasy listening indicator appears as a small icon on the TE Client screen whenever the speech-to-text engine is listening for input. The indicator is enabled or disabled from the emulation parameters for the TE Client.

To enable the Speakeasy listening indicator:

1 Access the Configuration Manager for the TE Client. For information on how to access the Configuration Manager, see the *Terminal Emulation Client User Guide*.

2 In the Indicators section, right-click the option **Speech-to-Text Listening Indicator** and select **Edit** from the context menu.

3 The *Speech-to-Text Indicator Light* dialog box appears. Select **Yes** from the drop-down menu and click **OK**.

When you use the Client to connect a session that uses speech-to-text, the listening indicator will appear whenever the Client is waiting for verbal input.

Speakeasy Settings

This section lists settings supported by Speakeasy. These settings can be modified using the screen reformatter or a script; some of them are modified using the microphone calibration or text-to-speech configuration. For detailed descriptions and value ranges, see the Speakeasy Settings section of the *Terminal Emulation Scripting Reference Guide*.

Text-to-Speech Settings

Setting	Description
tts_calibrate	Opens the speaker volume calibration wizard.
tts_external_speaker_setting	Speaker setting for use on Motorola/Symbol mobile devices.
tts_frequency	Indicates the sampling frequency.
tts_language_long	Displays the full name of the language currently being used.
tts_language_short	Displays the three-letter abbreviation of the language currently being used.
tts_pitch	Indicates the pitch level of spoken text.
tts_rate	Indicates the speed level.
tts_readmode	Indicates how text should be separated.
tts_voice	Indicates the name of the voice that is currently selected.
tts_volume	Indicates the sound level.
tts_waitfactor	Indicates the length of the pause between messages.

Speech-to-Text Settings

Setting	Description
stt_accuracy	This value affects the trade-off between CPU load, memory requirements, and accuracy.
stt_adjust_gain	This feature allows the engine to automatically increase and decrease the microphone input volume.
stt_beep_threshold	If the confidence value for a result is below this value, then a negative acknowledgement beep will not be played.
stt_calibrate	Opens the microphone calibration wizard.
stt_calibration_silence	Sets how long the user is expected to remain silent during a quick microphone calibration.
stt_confidence	Indicates the minimum difference in confidence required between the top two speech-to-text results for the top result to be accepted.
stt_expanded	Use this to get the confidence value along with the speech-to-text result.
stt_fx_detect_start	Indicates the action the speech engine should take before attempting to determine what the user is saying.
stt_fx_microphone	Tells the speech engine the distance between the user and the microphone.
stt_fx_min_duration	Indicates the minimum duration (in ms) of speech before speech detection is activated.
stt_fx_sensitivity	Indicates the speech detection sensitivity.
stt_fx_silence	Indicates the milliseconds of silence used to indicate the user is done speaking.
stt_fx_threshold	Indicates the amount of energy the microphone input must have before the speech detection is activated.

Setting	Description
stt_idle_timeout	Indicates the total milliseconds for the engine to continue collecting results following the last result or timeout.
stt_language_long	Displays the full name of the language currently being used.
stt_language_short	Displays the three-letter abbreviation of the language currently being used.
stt_logging	Creates a Speech-to-Text log file in the root folder of the device.
stt_logging_audio	Sets the engine to log speech-to-text attempts as .wav files.
stt_logging_engine	If set to 1, the speech-to-text engine will create a log file in the root folder of the device.
stt_preserve	Causes the speech engine to save the current engine state for use later.
stt_priority	Determines how aggressively the microphone input is collected and speech analysis is performed.
stt_processing	Indicates the action the speech engine should take when returning a grammar result.
stt_reset	Modifies engine adaptation speed and/or saved engine information.
stt_reset_session_delay	Indicates the total milliseconds for the speech engine to wait for a valid response before reverting back to the last saved state.
stt_result_sound	Causes a sound to play for result recognition.
stt_save_increase	Increases the threshold for saving a new engine state as time progresses.
stt_save_session_delay	Indicates the total milliseconds for the speech engine to wait before saving the next current state.

Setting	Description
stt_save_threshold	Directs the speech engine to save the state if the result confidence is greater than the result confidence for `stt_threshold` and `stt_save_threshold` combined.
stt_server_timeout	When uploading or downloading user training data, the value for this setting is how long (in seconds) the Client will wait for a response from the Avalanche server.
stt_size	Displays the size of the speech-to-text engine being used.
stt_special_sounds	Indicates how the speech engine should interpret special sounds.
stt_threshold	Indicates the minimum amount of confidence for the most-likely result that will be accepted.
stt_timeout	Indicates the total milliseconds (ms) for the system to wait before responding to the speaker.
stt_use_jumpback	Sets a buffer to check if the engine is processing speech.
stt_use_word_ids	Enables support for Word IDs (the !id directive) in grammar files.
stt_volume	Indicates the current volume of the microphone input.

Sample Speakeasy Scripts

This section contains example scripts that perform various Speakeasy functions. You can use the Script Editor to modify and customize scripts as desired. For more information on scripting or the Script Editor, see the *Terminal Emulation Scripting Reference Guide*.

Acquiring a Pick Quantity in the Industrial Browser

The following example script is designed to work with the Wavelink Industrial Browser and allows the user to speak a number 0-9999 (referred to as the pick quantity), ask for help, go back, disconnect the session, or clear the field.

This script requires that text-to-speech, speech-to-text, and a custom grammar file named `sapqty_four_digits.bnf` is installed. The script also calls a JavaScript function: clearFields. The text of the grammar file and the JavaScript function are included below.

This script calls another Wavelink script in order to repeat the information on the screen. The script `sapRepeatPrompts` is included in Changing Text-to-Speech Modes.

Acquiring a Pick Quantity Script

```
Comment: This function acquires the Pick Quantity via voice.
 Comment: Ensure Speakeasy support has been installed.
 If_Not( Speech_To_Text_Available )
 Ask_OK( "Speech-to-Text is not available.", "Error" )
 Return
 End_If
 If_Not( Speech_From_Text_Available )
 Ask_OK( "Text-to-Speech is not available.", "Error" )
 Return
 End_If
 Speech_To_Text_Cancel

Comment: The Speech-to-Text and Text-to-Speech languages are
specified.
 Speech_Change_Setting( "tts_language_short",
 Speech_Find_Setting_Value("tts_language_short","enu",FALSE))
 Speech_Change_Setting( "stt_language_short",
 Speech_Find_Setting_Value("stt_language_short","enu",FALSE))

Comment: Acquire Pick Quantity via Speech-To-Text.
 Comment: Initialize Speech-To-Text variables.
 bSpeechStarted = FALSE
 bSpeechDone = FALSE

Comment: Initialize confidence value. Speech uttered with
confidence values below this value will be rejected.
 Speech_Change_Setting( "stt_threshold", 4500 )
```

```
Comment: Start Speech-To-Text if not already started.
 Comment: This is needed so we start Speech_To_Text again if
nothing was stated before it times out.
 If_Not( bSpeechStarted )
 Speech_From_Text( "Enter Quantity:", TRUE )

Comment: With this Speech-To-Text function, the script
waits for voice input until stt_timeout value is reached if nothing is stated
sooner.
 bSpeechStarted = Speech_To_Text_No_Wait( bSpeechDone,
 strSpeechResult, "sapqty_four_digits" )

Comment: Wait_For_Screen_Update waits for speech as well.
 Wait_For_Screen_Update

End_If
 If( bSpeechDone )
 Comment: If sSpeechResult is not empty it signifies that we
received a speech result.
 Comment: Command and Control is handled here as well.

If_Not( String_Empty( strSpeechResult ) )

If( String_Equal( strSpeechResult, "repeat", 0, TRUE ) )

Comment: Set bRepeatPrompts so that the script
sapRepeatPrompts will be called.
 bRepeatPrompts = TRUE
 strSpeechResult = ""
 Speech_To_Text_Cancel
 Return
 End_If
 If( String_Equal( strSpeechResult, "back", 0, TRUE ) )

Comment: Press F2 key.
 Comment: Keypress_Key("VT220", "F2")
 Ask_OK( "F2 Key Pressed", "Back Function" )
 bBackFunction = TRUE
 strSpeechResult = ""
 Speech_To_Text_Cancel
 Return
 End_If
 If( String_Equal( strSpeechResult, "clear", 0, TRUE ) )
 Comment: Call clearFields javascript function
 Web_Scripting( "javascript:clearFields();" )
 strSpeechResult = ""
 Return
 End_If
 If( String_Equal( strSpeechResult, "quit", 0, TRUE ) )
 Comment: Disconnect session.
 Speech_To_Text_Cancel
 Disconnect
 Return
 End_If
 If( String_Equal( strSpeechResult, "help", 0, TRUE ) )
 Comment: Annunciate help list.
 Speech_To_Text_Cancel
 Speech_From_Text("Help list annunciates this list.", TRUE)
 Speech_From_Text("Repeat prompts repeats prompts and data.",TRUE)
 Speech_From_Text("Go back moves to the previous screen.",TRUE)
 Speech_From_Text( "Clear fields clears target bin, material, batch, and pick
quantity fields.", TRUE )
 Speech_From_Text("Quit SAP disconnects the session.", TRUE)
 strSpeechResult = ""
 Return
 End_If
 End_If
 End_If

Speech_To_Text_Cancel

Return
```

Acquiring Pick Quantity Grammar File

The following is an example of a grammar file that will recognize a four-digit number. This grammar file is used with the Acquiring a Pick Quantity script example.

```
#BNF+AM V1.0;

/*************************************************************

GRAMMAR: sapqty_four_digits.bnf

Description: This is a grammar that recognizes a four-digit number, 0 to
9999. It also recognizes the commands repeat, back, quit, and clear.

What Can I Say? You can say any number from 0 to 9999. You can also speak
two, three, or four separate digits.

************************************************************/

!grammar sapqty_four_digits;
!start <Speech>;

<Speech>: (edit | change quantity) <FourDigits> {@ = #2;} | <Command>;

<NonZeroDigit>: 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;

<Digit>:
 0 {@ = "0";} |
 OH {@ = "0";} |

<NonZeroDigit>;

<Teens>: 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19;

<Tens>:
 TEN {@ = "10";} |
 TWENTY {@ = "20";} |
 TWENY {@ = "20";} |
 THIRTY {@ = "30";} |
 THIRDY {@ = "30";} |
 FORTY {@ = "40";} |
 FORDY {@ = "40";} |
 FIFTY {@ = "50";} |
 FIFDY {@ = "50";} |
 SIXTY {@ = "60";} |
 SIXDY {@ = "60";} |
 SEVENTY {@ = "70";} |
 SEVENDY {@ = "70";} |
 EIGHTY {@ = "80";} |
 EIGHDY {@ = "80";} |
 NINETY {@ = "90";} |
 NINEDY {@ = "90";};

<CombinedTens>:
 TWENTY {@ = "2";} |
 TWENY {@ = "2";} |
 THIRTY {@ = "3";} |
 THIRDY {@ = "3";} |
 FORTY {@ = "4";} |
 FORDY {@ = "4";} |
 FIFTY {@ = "5";} |
 FIFDY {@ = "5";} |
 SIXTY {@ = "6";} |
 SIXDY {@ = "6";} |
 SEVENTY {@ = "7";} |
 SEVENDY {@ = "7";} |
 EIGHTY {@ = "8";} |
 EIGHDY {@ = "8";} |
 NINETY {@ = "9";} |
 NINEDY {@ = "9";};

<ExactThreeDigits>:

<NonZeroDigit> HUNDRED [AND] ( <Teens> | <CombinedTens> <NonZeroDigit> {@ =
#1+#2;} | <Tens> | <NonZeroDigit> {@ = "0"+#1;} ) {@ = #1+#4;} |

<NonZeroDigit> HUNDRED {@ = #1+"00";} |

<NonZeroDigit> ( <Teens> | <CombinedTens> <NonZeroDigit> {@ = #1+#2;} |
<Tens> ) {@ = #1+#2;} |

<Teens> {@ = "0"+#1;} | <CombinedTens> <NonZeroDigit> {@ = "0"+#1+#2;} |
<Tens> {@ = "0"+#1;} |

<Digit> {@ = "00"+#1;};

<FourDigits>:

<NonZeroDigit> THOUSAND [AND] <ExactThreeDigits> {@ = #1+#4;} |

<NonZeroDigit> THOUSAND {@ = #1+"000";} |

<NonZeroDigit> HUNDRED [AND] ( <Teens> | <CombinedTens> <NonZeroDigit> {@ =
#1+#2;} | <Tens> | <NonZeroDigit> {@ = "0"+#1;} ) {@ = #1+#4;} |

<NonZeroDigit> HUNDRED {@ = #1+"00";} |
<NonZeroDigit> ( <Teens> | <CombinedTens> <NonZeroDigit> {@ = #1+#2;} |
<Tens> ) {@ = #1+#2;} |

<Teens> |<CombinedTens> <NonZeroDigit> {@ = #1+#2;} | <Tens> |

!repeat(<Digit> {@ = previous.@ + #1;},1,4);

<Command>: repeat prompts {@ = "repeat";} |

go back {@ = "back";} | quit sap {@ = "quit";} |

clear fields {@ = "clear";} | help list {@ = "help";};
```

JavaScript Function clearFields

The following JavaScript function is an example of a function that will clear the fields in the Wavelink Industrial Browser. This is used with the Acquiring a Pick Quantity script example.

```
/**************************************************
This function is called to clear the fields when the clear command and
control command is spoken.

There are no inputs provided when this function is called.
**************************************************/

function clearFields()
 {
 //Clear target Bin field
 var strFieldNameClear = "s4000_binp[1]";
 var strFieldValueClear = "";
 document.getElementsByName(strFieldNameClear).value = strFieldValueClear;
 var arrayFieldNameClear = document.getElementsByName(strFieldNameClear);
 arrayFieldNameClear[0].value=strFieldValueClear;

//Clear target Material field
 var strFieldNameClear = "s4000_matnrp[1]";
 var strFieldValueClear = "";
 document.getElementsByName(strFieldNameClear).value = strFieldValueClear;
 var arrayFieldNameClear = document.getElementsByName(strFieldNameClear);
 arrayFieldNameClear[0].value=strFieldValueClear;

//Clear target Batch field
 var strFieldNameClear = "s4000_chargp[1]";
 var strFieldValueClear = "";
 document.getElementsByName(strFieldNameClear).value = strFieldValueClear;
 var arrayFieldNameClear = document.getElementsByName(strFieldNameClear);
 arrayFieldNameClear[0].value=strFieldValueClear;

//Clear Pick qty field
 var strFieldNameClear = "s4000_qty[1]";
 var strFieldValueClear = "";
 document.getElementsByName(strFieldNameClear).value = strFieldValueClear;
 var arrayFieldNameClear = document.getElementsByName(strFieldNameClear);
 arrayFieldNameClear[0].value=strFieldValueClear;

}
```

Changing Text-to-Speech Modes

When you use text-to-speech, the mode determines if it reads character strings as words or pronounce each letter separately. The text-to-speech mode is configured using the tts_ readmode setting. The following example script demonstrates how to change the text-to-speech mode working in a voice-picking environment.

The following script also repeats picking information from the screen when the user requests it. This script is named `sapRepeatPrompts` and is designed to work with the script available in Acquiring a Pick Quantity in the Industrial Browser.

```
Script( sapRepeatPrompts )
String( strPromptBin )
String( strPromptBinValue )
String( strPromptMaterial )
String( strPromptMaterialValue )
String( strPromptBatch )
String( strPromptBatchValue )
String( strPromptQuantity )
String( strPromptQuantityValue )
String( strSearchString )
Number( nReadMode )
Number( nIndex )

Comment: This function repeats the prompts when called from Command and
Control.
 Speech_To_Text_Cancel

Comment: The Speech-to-Text and Text-to-Speech languages are specified.
 Speech_Change_Setting( "tts_language_short", Speech_Find_Setting_Value("tts_
language_short", "enu", FALSE))
 Speech_Change_Setting( "stt_language_short", Speech_Find_Setting_Value(
"stt_language_short", "enu", FALSE))

Comment: Ensure read mode is in sentence mode.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "sentence", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annunciate Bin prompt.
 Speech_From_Text( strPromptBin, TRUE )

Comment: Change read mode to character mode to annunciate bin number.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "character", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annuciate Bin number.
 Speech_From_Text( strPromptBinValue, TRUE )

Comment: Change read mode back to sentence mode to annunciate material
prompt.
```

```
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "sentence", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annunciate Material prompt.
 Speech_From_Text( strPromptMaterial, TRUE )

Comment: Change to character mode to annunciate material.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "character", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annuciate Material.
 Speech_From_Text( strPromptMaterialValue, TRUE )

Comment: Change to sentence mode to annunciate batch prompt.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "sentence", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annunciate Batch prompt.
 Speech_From_Text( strPromptBatch, TRUE )

Comment: Change to character mode to annunciate Batch.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "character", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Annuciate Batch.
 Speech_From_Text( strPromptBatchValue, TRUE )

Comment: Change to sentence mode to annunciate pick quantity.
 nReadMode = Speech_Find_Setting_Value( "tts_readmode", "sentence", FALSE )
 Speech_Change_Setting( "tts_readmode", nReadMode )

Comment: Extract quantity value left of the period.
 strSearchString = "."
 nIndex = String_Find_First( strPromptQuantityValue, strSearchString, TRUE )
 strPromptQuantityValue = String_Left( strPromptQuantityValue, nIndex )
 strPromptQuantityValue = String_Trim_Spaces_Start( strPromptQuantityValue )

Comment: Annunciate Pick Quantity prompt.
 Speech_From_Text( strPromptQuantity, TRUE )

Comment: Annunciate Pick Quantity Value.
 Speech_From_Text( strPromptQuantityValue, TRUE )

Return
```

Displaying Speech Results in a Dialog Box

The following example script prompts the user for a number, converts the spoken number into text, and displays it in a dialog box on the mobile device. This script requires the `connected_digits` grammar file.

```
String(sResult)
Speech_From_Text("Say a number",FALSE)
Speech_To_Text(sResult, "connected_digits")
Ask_OK(sResult,"Number Returned")
Return
```

In this example the number is treated as a string because both the functions Speech_To_Text and Ask_OK require a string. However, it could be converted to a number if needed by using the following line:

```
nResult = String_To_Number_Decimal (sResult)
```

Reading the Screen Aloud

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

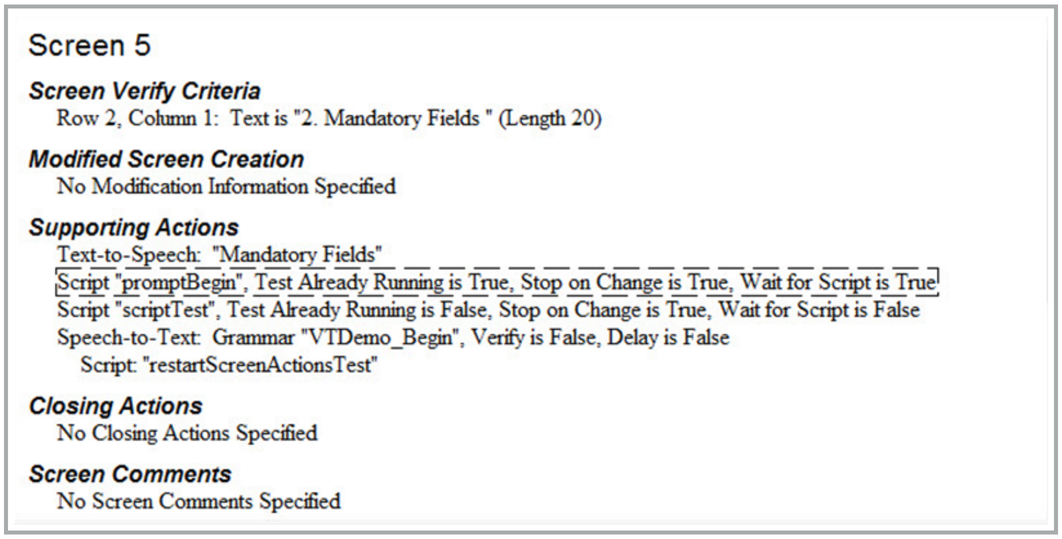
```
nNumRows=Get_Screen_Rows
nCurrentRow=1
While(Number_Less_Than_Or_Equal(nCurrentRow,nNumRows))
 Speech_From_Text(Get_Screen_Text(nCurrentRow,1),FALSE)
 nCurrentRow=Number_Plus(nCurrentRow,1)
End_While
Return
```

Restart Supporting Action

The script function, Restart_Supporting_Actions, provides a way for the user to restart the supporting actions on a given reformatted screen. It also allows the user to specify a supporting action index as the new starting point. The following is an example demonstrating how the function can be used:

```
Script(restartScreenActionsTest)
Boolean(bCancelScripts, True)
Number(nStartAction, True)
 Comment: Restart supporting actions.
 Restart_Supporting_Actions( nStartAction, bCancelScripts)
 Message("Restarted Supporting Actions.", 5)
Return
```

`bCancelScripts` is a True/False parameter that determines if any actions currently working should be stopped immediately. When set to True, it will stop any actions currently working that are set to recycle. If set to false, all supporting actions will be stopped, except for script actions. Script actions continue to use behavioral options selected in the script dialog.


Screen 5

Screen Verify Criteria
Row 2, Column 1: Text is "2. Mandatory Fields " (Length 20)

Modified Screen Creation
No Modification Information Specified

Supporting Actions
Text-to-Speech: "Mandatory Fields"
Script "promptBegin", Test Already Running is True, Stop on Change is True, Wait for Script is True
Script "scriptTest", Test Already Running is False, Stop on Change is True, Wait for Script is False
Speech-to-Text: Grammar "VTDemo_Begin", Verify is False, Delay is False
Script: "restartScreenActionsTest"

Closing Actions
No Closing Actions Specified

Screen Comments
No Screen Comments Specified

`nStartAction` is a numeric parameter index that determines what supporting action to start. Entering 0 or 1 will start at the first supporting action.

Speech Demo Script

The following example script creates the four buttons on the screen: **Digits**, **State**, **Play Screen**, and **Done**. The **Digits** and **State** buttons allow the user to input a verbal response which is then displayed on the screen. The **Play Screen** button causes the mobile device to read back all the text on the screen, and the **Done** button allows the user to exit the script.

```
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)

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
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
```

Use WAV File as Voice Input

The following example script assists the user in automated testing, allowing for a recorded input to be used as a recognized voice input source for Speakeasy. When using the full calibration wizard, you can create WAV recordings that can be played back against the grammar files that contain the recorded content.

```
Script(useWavFileAsVoiceInput)
Activate(From_Menu)
Speech_To_Text_Set_Audio_Test_File("c:\\speakeasy\\FSE-North-America-
Training_5-1-14\\STT1.wav", TRUE, FALSE)
Speech_Change_Setting("stt_result_sound", 0)
Return
```

The absolute path to the WAV file must be provided, or the engine will resume receiving speech input from a microphone. Following the path are two True/False parameters, the first determining if the file should be restarted when a grammar change takes place, and the second determining if the audio file should be played aloud from the device's speakers as it is supplied to Speakeasy for voice input. If the second parameter is set to FALSE, it will still play against Speakeasy's grammar files for the purpose of improved automation testing.

Using a Voice Profile

The following example script asks the user for his username and then switches to the voice profile associated with that username.

```
Script( Set_Speech_User )
String( sSpeechToTextUserName, True )
Activate( Connection )
 Comment: If the persistent variable doesn't have a value, the default is the
username the Speech-to-Text engine is using.
 If( String_Empty( sSpeechToTextUserName ) )
 sSpeechToTextUserName = Speech_To_Text_Get_User_Name
 End_If

Comment: Ask the user the username to use. The last username used will be the
default.
 sSpeechToTextUserName = Ask_String_Lowercase( "What is your Speech
Username?", "Speech", 1, 100, sSpeechToTextUserName )

Comment: Tell the Speech-to-Text engine the username.
 Speech_To_Text_Change_User_Name( sSpeechToTextUserName )
 Return
```

Wavelink Contact Information

To provide feedback about the Avalanche help, click the **Email Feedback** button on the toolbar above.

For product downloads or documentation, go to the Wavelink downloads page:

http://www.wavelink.com/download-software

For information on contacting Wavelink, please go to:

http://www.wavelink.com/Customer-Care-Contact-Customer-Care