
Migration Guide



S P E E C H I F Y 3 . 0

Document History

Date	Release Name
December 2003	First Edition, update 2 — for Speechify 3.0
November 2003	First Edition, update 1 — for Speechify 3.0
August 2003	First Edition — for Speechify 3.0

Notice

Copyright © 2000–2003 by ScanSoft, Inc. All rights reserved.

The information in this document is subject to change without notice.

Use of this document is subject to certain restrictions and limitations set forth in a license agreement entered into between SpeechWorks International, Inc. and purchaser of the Speechify software. Please refer to the SpeechWorks license agreement for license use rights and restrictions.

SpeechWorks is a registered trademark, and OpenSpeech, DialogModules, SMARTRecognizer, Speechify, Speechify Solo, SpeechSecure, SpeechSite, SpeechWorks Here, the SpeechWorks logo and the SpeechWorks Here logo are trademarks or registered trademarks of SpeechWorks International, Inc. in the United States and other countries. All other trademarks are property of their respective owners. Windows NT is a registered trademark of Microsoft Corporation.

Portions of Speechify software are subject to copyrights of GlobeTrotter Software, Inc.

Published by:

ScanSoft, Inc.
Worldwide Headquarters
9 Centennial Drive
Peabody, MA 01960
United States

Table of Contents

Migrating systems from Speechify 2.x to 3.x v

I. Required migration tasks I

Install the new server	1
Install new voice packages	1
Acquire licenses from ScanSoft.....	2
Update handling of error codes	2

2. Required in some instances 3

Upgrade operating system	3
Change the US English “Rick” voice to the “Tom” voice	3
Upgrade W3C SSML support to LCWD of December 2002	4
Check assumptions about “state” for embedded tags	5
Change scripts/tools that control the Speechify Windows service	6
Update event log scripts	6
Change SAPI integrations that “fake” the language of certain voices	6
Review scripts/tools that process Speechify diagnostic/error logs	7
Check assumptions about the real-time audio generation rate	7

3. Recommended migration tasks 9

Test your applications	9
Remove application/integration support for W3C SSML <audio>	9
Review use of deprecated dictionary modification API functions	10
Watch use of existing dictionaries for errors	10
Change server startup scripts	10
Use SWIttsOpenPortEx() instead of SWIttsOpenPort()	11
Use new feature for stricter W3C SSML validation	11
Change code that adjusts audio packet sizes	11
Update all dictionaries to new format	12
Old proprietary format	13
New XML format	13
Specify dictionary locations explicitly	13
Review use of deprecated SWIttsPing() function	13
Change code to stop referencing SWIttsMessagePacket for SWItts_cbError	14

4. Miscellaneous migration issues	17
Watch-dog system removed.....	17
Logging to operating system logs is now on by default	17
The Speechify server is now multi-threaded, not multi-process	17
Review tools that depend on specific audio output	18
The default installation path for Speechify has changed	18
The GUI demo application has changed on Windows	18
 Index.....	 19



Migrating systems from Speechify 2.x to 3.x

This document discusses topics for platform and application developers who migrate existing Speechify 2.x systems to the 3.0 release. If you are an end user who purchased a system that uses Speechify through a platform provider, please contact your platform provider for information on migrating. The topics are described from several perspectives:

- ❑ “Required migration tasks” on [page 1](#) – issues that must be addressed.
- ❑ “Required in some instances” on [page 3](#) – issues that might need to be addressed.
- ❑ “Recommended migration tasks” on [page 9](#) – issues for review and consideration.
- ❑ “Miscellaneous migration issues” on [page 17](#) – issues that typically require no action.

Overview

Speechify 3.0 adds new features and simplifies existing features without requiring code changes to systems that use Speechify 2.x. However, there are required and recommended migration tasks.

New features and technical details

Because this guide focuses on migration issues, it limits the discussion of new features to the migration tasks to be performed. For an explicit list of new features, see the release notes or consult the Speechify 3.0 marketing literature.

The ScanSoft product documentation covers every topic discussed in this migration guide. See the table of contents and index in each book:

- ❑ *Speechify User's Guide*
- ❑ *SpeechWorks Licensing Handbook*
- ❑ *Speechify Language Supplement* (one per language)



Required migration tasks

Install the new server

At a minimum, you need to install the Speechify 3.0 server component when upgrading existing installations. On Linux, this is contained in the file `Speechify-Engine-3.0-0.i386.rpm`. On Windows, this is the component named “Speechify server run-time” in the installer. See “Installing and Configuring Speechify” in the *Speechify User’s Guide* for details.

There are two levels of compatibility with Speechify clients.

1. The Speechify 3.0 server is designed to work with existing Speechify 2.1 clients. Applications that use a Speechify 2.1 client should work without modification. Some of the new features in the Speechify 3.0 server are also available to existing Speechify 2.1 client applications via tags and elements in the input text. See the *Speechify User’s Guide* for more details.
2. If you upgrade to the Speechify 3.0 client, you get access to the full set of new features available to applications. The Speechify 3.0 client also supports more robust networking by maintaining a configurable heartbeat with the server.

Install new voice packages

All of the Speechify voices have been repackaged in a new format for 3.0. You must upgrade and reinstall all voices that you use with the new packages provided by ScanSoft. See “Installing and Configuring Speechify” in the *Speechify User’s Guide* for details.

Acquire licenses from ScanSoft

Speechify 3.0 now requires run-time licenses in order to execute. The license enforcement system is installed as part of the Speechify installation procedure. You need one license per instance of a TTS engine. Request your licenses by sending e-mail to techsupport@speechworks.com. The *SpeechWorks Licensing Handbook* (SWILicensingHandbook.pdf) is located in the doc subdirectory of your installation directory. Please read this document, as it covers all information about license enforcement.

Update handling of error codes

Speechify 3.0 can return new error codes to the client application. Platform developers and any application developers who access the SWItts API functions must update their handling of return values. You can reference SWItts.h for the enumerations; your code should handle every possible error code (and, for future forward compatibility, should include a default case to handle unexpected return values). For details on error code meanings, see the *Speechify User's Guide*.

New codes that apply to Speechify 2.1 and 3.0 client applications:

- ❑ SWItts_ALREADY_INITIALIZED
- ❑ SWItts_NO_LICENSE

New codes that only apply to Speechify 3.0 client applications:

- ❑ SWItts_DICTIONARY_ACTIVE
- ❑ SWItts_DICTIONARY_INVALID_PRIORITY
- ❑ SWItts_DICTIONARY_INVALID_TYPE
- ❑ SWItts_DICTIONARY_LOADED
- ❑ SWItts_DICTIONARY_NOT_LOADED
- ❑ SWItts_DICTIONARY_PARSE_ERROR
- ❑ SWItts_DICTIONARY_PRIORITY_ALREADY_EXISTS
- ❑ SWItts_INVALID_MEDIATYPE
- ❑ SWItts_LICENSE_ALLOCATED
- ❑ SWItts_LICENSE_FREED
- ❑ SWItts_UNSUPPORTED
- ❑ SWItts_URI_FETCH_ERROR
- ❑ SWItts_URI_NOT_FOUND
- ❑ SWItts_URI_TIMEOUT



Required in some instances

Many of the following tasks are required for migration depending on which Speechify 2.x features your system uses.

Upgrade operating system

The *Speechify User's Guide* describes the supported operating systems for this release. Consult the user's guide and upgrade your server if needed.

Change the US English “Rick” voice to the “Tom” voice

ScanSoft no longer supports the US English male voice named Rick. Applications that use Rick should now use the voice named Tom instead. Contact ScanSoft Technical Support or your sales representative for information on obtaining the Tom voice.

Upgrade W3C SSML support to LCWD of December 2002

Speechify 2.x supported the January 2001 draft of the W3C Speech Synthesis Markup Language (W3C SSML) standard. Speechify 3.0 now supports a more recent draft: the Last Call Working Draft of December 2002. If you use W3C SSML with Speechify 2.x, there are two possible migration paths:

- ❑ Migrate all of your W3C SSML documents/output to this version of the standard.
- ❑ Set the Speechify configuration parameter `tts.ssml.doStrictValidation` to false (the default is true).

Using the latter technique is a stop-gap measure; ScanSoft reserves the right to fully remove support for the January 2001 W3C SSML draft standard in a future release. Also, many warning messages have been added for unsupported constructs (attributes, say-as types, etc.). The audio generated in these cases remains the same but you may see warnings that you did not see in Speechify 2.x.

Here is an overview of the changes in W3C SSML behavior between Speechify 2.1 and Speechify 3.0:

Supported elements:

Element name	Notes
break	Support added for “x-small” and “x-large” sizes.
desc	Supported, but Speechify does not provide access to the description text.
lexicon	Dictionaries must be in the SpeechWorks XML dictionary format.
metadata	Supported, but Speechify does not provide access to the metadata.
prosody	Support added for named rate values “x-fast” (200%) and “x-slow” (33%). Support added for named volume values “x-soft” (16%) and “x-loud” (150%).
say-as	While W3C SSML defines the “interpret-as,” “format,” and “detail” attributes, their possible values are not standardized. Speechify supports a wide variety of values; see the special section below for details.
sub	Equivalent to <code><say-as sub=“...”></code> in Speechify 2.x.
voice	Partial support added. The “age” and “variant” attributes are ignored. Switching voices and languages is not supported, so if the “xml:lang,” “gender,” and/or “name” attribute do not match the current Speechify server voice, a warning is logged to the Speechify error log with the contained text simply spoken in the current Speechify server voice.

Support for the “say-as” element:

Attribute	Notes
Acronym	Using detail="strict" results in speaking punctuation (such as speaking commas as “comma”).
Cardinal	Supported if relevant in the target language.
Digits	Supported if relevant in the target language.
Letters	Letters and numbers are pronounced individually. Using detail="strict" results in speaking punctuation (such as speaking commas as “comma” – same as "acronym".)
Number	Support added for the “telephone” format attribute. Roman numerals are not supported. All the format values are supported as interpret-as values as well, behaving the same for either syntax.
Ordinal	Supported if relevant in the target language
Telephone	Using detail="punctuation" results in speaking punctuation (such as speaking dashes as “dash”).
Words	Support added.

Check assumptions about “state” for embedded tags

In Speechify 2.x, some embedded tags retained “state.” That is, if the input text for a speak request used one of these tags, the TTS engine would not reset once the speak request finished and the next speak request would be affected by the previous change. In Speechify 3.0, this behavior has changed and all tag states are reset after each speak request. This enhancement makes it simpler for users in dynamically switched or multiplexed designs to implement Speechify. The affected tags are:

Spellout tags: \!tsa, \!tsc, \!tsr, \!ts0
 Number/year tags: \!ny1, \!ny0
 Postal address tags: \!addr1, \!addr0
 Currency tags (es-MX): \!cdd, \!cdp

Check all of the text fed into Speechify for these tags, and ensure that the corresponding end tag is specified within the same block of text. If it is not, Speechify 3.0 may generate different audio than originally designed; the application may be relying on tags to carry over between text blocks.

Change scripts/tools that control the Speechify Windows service

In Speechify 2.x, Speechify had one Windows service named “Speechify.” In Speechify 3.0, each installed voice is a unique service. If you wrote a script or code to programmatically control the service, you need to review and revise this script. For example, in a batch file, “net start Speechify” no longer works. If you have the 8 kHz version of the US English Tom voice installed, you can start that voice with the command “net start SpfyTom8”.

Update event log scripts

This task might be required for platform or application developers who have created scripts to extract or display information from the Speechify event log. Speechify 3.0 adds both new events and new tokens to existing events. There are new events that describe license acquisitions and releases, dictionary and audio cache hits and misses, internet fetches, and logging of generated pronunciations. In addition, the existing STRT event has new tokens. See “Speechify Logging” in the *Speechify User’s Guide* for more detailed information.

Change SAPI integrations that “fake” the language of certain voices

Speechify 2.x supports SAPI 5 but not the SAPI 5 Universal Phone Set (UPS). This means that Speechify 2.x offered 100% support only for the original six languages that were shipped with SAPI 5: US English, Japanese, Mandarin Chinese, (French) French, Castilian Spanish, and (German) German. Languages that are not variants of those listed, such as Brazilian Portuguese, did not work with Speechify and SAPI. Languages that are close variants of those listed, such as Mexican Spanish and British English, could be made to work (mostly) if SAPI was told that they were one of the six languages listed above. Therefore, many Speechify users modified their installations so that SAPI recognized these variants as one of the core SAPI languages.

Speechify 3.0 adds support for the UPS. With this addition, Speechify voices advertise themselves as the actual language variant that they are and applications should request them as such. Any applications that requested the voice as one of the core six languages may need to be changed in order to use the voice with Speechify 3.0.

Review scripts/tools that process Speechify diagnostic/error logs

If you wrote a script or code to interpret Speechify logs, you need to review and revise this script. All error ID numbers have changed and have corresponding changes to the SpeechifyErrors.en-US.xml file. The basic format for a line in the error/diagnostic logs has changed. The fields on each line may have changed completely or just changed order. Here are the changes for the diagnostic logs:

2.1:

```
<timestamp>|<OS process ID>|<tag ID>|<message>
```

3.0:

```
<timestamp>|<OS thread ID>||<logical channel  
ID>|<tagID>|<message>
```

Here are the changes for the error logs:

2.1:

```
<timestamp>|<OS process ID>|<error ID>|<key/value pairs>
```

3.0:

```
<timestamp>|<OS thread ID>||<logical channel  
ID>|<CRITICAL|SEVERE|WARNING|INFO>|<error ID>|<error ID  
text>|<key/value pairs>
```



NOTE

The “||” in the 3.0 examples is intentional. Empty fields exist for log format compatibility with SpeechWorks OpenSpeech Browser.

Check assumptions about the real-time audio generation rate

When predicting the performance of applications, do not rely on extrapolations of the real-time audio generation rate as measured under light loads. In Speechify 3.0, the generation rate is throttled by the engine and does not provide a measurement that can be scaled.

Background information:

- ❑ In Speechify 2.x, the real-time audio generation rate could often be very high (10 to 50 times real-time) when only a few server ports were active. The Speechify 3.0 server throttles audio generation in this situation so it never exceeds two times real-time for any given synthesis operation. This means that the real-time audio rate delivered by a Speechify 3.0 server under light loads (a few simultaneous synthesis operations) is almost always lower than the real-time rate of Speechify 2.1.
- ❑ However, the Speechify 3.0 server also consumes less CPU and memory, which results in the ability for a single computer to support many more active ports and can accommodate much heavier loads under real-world scenarios (more simultaneous synthesis operations), or can accommodate running other application processes on the server machine.

Rather than predicting application performance based on measuring the real-time audio generation rate under light loads, you should check the performance test procedures and criteria that you use for verifying your application. This allows you to verify performance based on the actual port density you can achieve (i.e. running a large number of application instances).



Recommended migration tasks

Test your applications

The Speechify 3.0 release is designed to be fully backward compatible. However, it is not possible to foresee all possible interactions between Speechify, platform, and application code. ScanSoft recommends a complete regression test of applications after upgrading from 2.x to 3.0.

Remove application/integration support for W3C SSML <audio>

Speechify 3.0 now adds full support for the <audio> element of the W3C SSML standard. Users with VoiceXML integrations of Speechify can reduce their platform's complexity by delegating all VoiceXML prompting to Speechify, including fetching and playing pre-recorded prompts. To do so, your platform should:

- ❑ extract the VoiceXML prompt requests into an W3C SSML document:
 - use `xml:base` on the <speech> element to indicate the base URL for relative URL handling
 - resolve <audio> `expr` attributes and insert the results into the W3C SSML document as <audio> `src` attributes
- ❑ send the entire W3C SSML document to Speechify for playback

Speechify fetches the pre-recorded prompts that are intertwined with text-to-speech and returns one integrated audio stream, instead of the integration having to manage audio fetches, conversions, and insertions.

Review use of deprecated dictionary modification API functions

Speechify 3.0 supports the set of dictionary API functions that were available in Speechify 2.x, however, those API functions are now deprecated and are not recommended for use in new Speechify applications. The functions remain and are supported for backwards compatibility but may be removed in a future release of Speechify. These include:

- ❑ SWIttsAddDictionaryEntry()
- ❑ SWIttsDeleteDictionaryEntry()
- ❑ SWIttsResetDictionary()
- ❑ SWIttsLookupDictionaryEntry()
- ❑ SWIttsGetDictionaryKeys()

Speechify 3.0 has a new set of API functions for manipulating dictionaries at the file level instead of entry-by-entry. These functions allow entire dictionary files to be fetched and activated. This design assumes that distinct tools let application developers modify these dictionaries entry-by-entry. See the *Speechify User's Guide* for information on the Speechify 3.0 dictionaries and API functions.

Watch use of existing dictionaries for errors

Speechify 3.0 now performs tighter error checking on dictionary entries. Speechify 2.x silently ignored errors, leading to subtle, hard-to-detect problems. Speechify 3.0 now logs these errors to the system's error log. Also, Speechify 3.0 now logs erroneous SPRs embedded in input text.

Change server startup scripts

Command-line arguments from Speechify 2.x are supported for backward compatibility but are now deprecated and are not recommended for use in new scripts. The preferred mechanism to control parameters in Speechify 3.0 is via XML-based configuration files.

Use SWIttsOpenPortEx() instead of SWIttsOpenPort()

Speechify 3.0 adds a new API function named SWIttsOpenPortEx(). The purpose of this new function is to bring the Speechify and Speechify Solo APIs in synch so that applications written for one work with minimal effort for the other. The next release of Speechify Solo will add this function as well. SWIttsOpenPort() remains supported but simply maps to a call to SWIttsOpenPortEx(). ScanSoft may remove the SWIttsOpenPort() API in a future release. See “API Reference” in the *Speechify User’s Guide* for more information.

Use new feature for stricter W3C SSML validation

Speechify 3.0 now offers the ability to do stricter validation of W3C SSML. This feature is enabled/disabled by changing the value of the tts.ssml.doStrictValidation parameter in the Speechify configuration file. This option is enabled by default. You can use this to ensure that your W3C SSML documents are error-free and thus produce the expected output.

For comparison, in Speechify 2.x, the SWIttsSpeak() function only returns SWItts_SSML_PARSE_ERROR if the input W3C SSML document is not well formed XML. If the W3C SSML contained in that document had errors, Speechify would ignore unknown elements and attributes and speak the contained text.

In Speechify 3.0, with strict parsing turned off, the output is similar to Speechify 2.x but warnings are logged for elements/attributes/attribute values that are not understood.

In Speechify 3.0, with strict parsing turned on (the default), a W3C SSML document containing errors causes the SWIttsSpeak() function to fail with a return code of SWItts_SSML_PARSE_ERROR. Errors are logged on the server with reasons and line numbers.

Change code that adjusts audio packet sizes

In Speechify 2.x, applications could set the audio packet size via tts.network.packetsize to a multiple of 1K (1K, 2K, 4K, or 8K) or a multiple of the network MTU. While these K and MTU based values are retained for backward compatibility in Speechify 3.0, these are now deprecated and will be removed in a

future release of Speechify. (K-based values are fully honored, but MTU-based sizes are ignored with a packet size of 4096 bytes used instead.) In addition, the parameter name has been renamed; the old name remains as an alias for backward compatibility, but the old name will be removed in a future release of Speechify as well.

Instead, applications should now set `tts.audio.packetsize`, specifying a size in bytes, where Speechify now supports any even byte size value between 64 bytes and 102400 bytes. (The Speechify 3.0 server has been changed so that it self-optimizes the TCP/IP network packets sent to the client independent of the audio packet size delivered to the application callback, so using MTU-based values is no longer important for optimizing TCP/IP network transport. Instead, set `tts.audio.packetsize` purely based on the application's size preference for audio delivered to the application callback.)

Update all dictionaries to new format

Application developers should create all new pronunciation dictionaries using XML tags instead of using the proprietary syntax used in Speechify 2.x. The new syntax is described in detail in the *Speechify User's Guide*. Speechify 3.0 provides an automated conversion utility named `SWIttsMigrateDictToXML`. The purpose of this change, aside from the added power of the XML format, is to align Speechify dictionaries with the OpenSpeech Recognizer dictionaries to enable seamless integrations between the two in the future.

`SWIttsMigrateDictToXML` is a command-line utility with these options:

Option	Description
<code>-i</code>	Use <inputfile> as the input Speechify 2.1 dictionary. Default: stdin.
<code>-o</code>	Use <outputfile> as the output Speechify 3.0 dictionary. Default: stdout.
<code>-l</code>	Use <language>. Default: "en-US".
<code>-t</code>	Use dictionary type <dicttype>. Default: "main".
<code>-, -h</code>	Show command line usage information

For example:

```
SWIttsMigrateDictToXML -l en-US < en-US-MainESPR-21.dic >
en-US-MainESPR-30.xml
SWIttsMigrateDictToXML -i en-US-21.dic -o en-US-30.xml -l
en-US -t main
SWIttsMigrateDictToXML -i ja-JP-mainext-21.dic -o ja-JP-
mainext-30.xml -l ja-JP -t mainext
```

Old proprietary format

```
tomato      \![tXmeto]
george      \![JcrJ]
```

New XML format

This example contains a superset of the information shown in the old format above.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE lexicon PUBLIC "-//SpeechWorks//DTD LEXICON 1.0//
  EN" "lexicon.dtd">
<lexicon xml:lang="en-US" type="main" alphabet="text">
  <entry key="tomato">
    <definition value="tXmeto" />
  </entry>
  <entry key="george">
    <definition value="JcrJ" />
  </entry>
```

Specify dictionary locations explicitly

Speechify 2.x automatically loads a language-specific dictionary and then a voice-specific dictionary, both from pre-defined locations in the Speechify installation dictionary. This is still supported in Speechify 3.0, but Speechify 3.0 supports a much more flexible way of configuring dictionaries. You can tell Speechify to read your dictionaries from any URI (e.g. file:// for local disk and http:// to fetch from a web-server). You can tell Speechify to read these dictionaries by using the new `SWIttsDictionaryLoad()` function, by listing them in the `tts.engine.dictionaries` parameter in the Speechify configuration file, or by using a new embedded text tag.

Review use of deprecated `SWIttsPing()` function

The `SWIttsPing()` function in Speechify 2.x was intended to let the application verify connectivity between the Speechify client and the Speechify server. Although still supported in Speechify 3.0 for backward compatibility, applications should be

changed to no longer use this function. The Speechify 3.0 client now automatically performs heart-beating with the server to verify connectivity. ScanSoft may remove this function completely from a future release.

Change code to stop referencing SWIttsMessagePacket for SWItts_cbError

In earlier releases, the SWItts_cbError event served two purposes:

1. Notify the application that a fatal port error occurred and that it must close the indicated port (and typically reopen it after the close completes)
2. Provide information on the failure so the integration could optionally notify the system operator using the native application logging system

SWItts_cbLogError provided failure information in the same way, but unlike SWItts_cbError the port remained valid and the application did not need to close it. Because these events were similar from an application logging perspective but different from an application error handling perspective, developers could get confused.

To resolve the confusion, Speechify 3.0 includes the following change:

1. SWItts_cbLogError is used to report information about both fatal and non-fatal errors. When this event is received, the integration may simply opt to log the message to the system operator. The application should not take any other action for this event (should not close the port).
2. In the case of a fatal port error, a SWItts_cbError event is delivered immediately following the SWItts_cbLogError event. When this event is received, the integration must close the port. It should not examine the SWIttsCallback event data pointer, which is now documented as being a NULL pointer.

To migrate this behavior, change your applications to access and log the SWIttsMessagePacket structure for SWItts_cbLogError events only, instead of both SWItts_cbLogError and SWItts_cbError events. In other words, the application event handling is simplified so SWItts_cbLogError only relates to logging, while SWItts_cbError only relates to closing and re-opening failed ports.



NOTE

For compatibility, SWItts_cbError still delivers a SWIttsMessage packet, but

that information is redundant with the preceeding SWItts_cbLogError message, resulting in duplicate information if the application logs both. A future version of Speechify may start delivering a NULL pointer for SWItts_cbError, causing any application that skips this migration task to crash.



Miscellaneous migration issues

Watch-dog system removed

Speechify 2.x had a watch-dogging system to maintain the pool of server processes on Windows. This has been removed since Speechify 3.0 is now multi-threaded. All of the command-line parameters that controlled the watch-dogging parameters are still accepted but are now ignored with a warning logged.

Logging to operating system logs is now on by default

To make Speechify 2.x log errors to the operating system's log (syslog on Unix, Event Log on Windows), the server required the `--system_log` option on the command line. In Speechify 3.0, this is now the default behavior. This feature is enabled/disabled by changing the value of the `tts.log.diagnostic.toSystemLog` parameter in the Speechify configuration file.

The Speechify server is now multi-threaded, not multi-process

In Speechify 2.x, the Speechify server was multi-process, i.e., the list of running processes would show multiple instances of the Speechify process. The Speechify 3.0 server is now multi-threaded and a single Speechify process contains multiple

instances of the same engine. If you have written any operational tools or scripts that monitor the operating system's task list and that rely on the previous behavior, you need to modify those tools.

Review tools that depend on specific audio output

If you have tools that compare output from Speechify to a “gold master” test set by doing a binary comparison, you need to generate a new gold test set.

The default installation path for Speechify has changed

This table shows the default installation path for Speechify software:

Operating system	2.x default	3.0 default
Windows	C:\Program Files\Speechify	C:\Program Files\SpeechWorks\Speechify
Linux	/usr/local/Speechify'	/usr/local/SpeechWorks/Speechify

The GUI demo application has changed on Windows

Speechify 2.x shipped with a sample application named “dialogGUI.” In Speechify 3.0 this is replaced by a new sample named “spfyDemo.” This demo offers more functionality, including streaming output and the ability to change the speaking rate.

Index

Symbols

\!addr0 2-5
\!addr1 2-5
\!cdd 2-5
\!cdp 2-5
\!ny0 2-5
\!ny1 2-5
\!ts0 2-5
\!tsa 2-5
\!tsc 2-5
\!tsr 2-5

A

acronym say-as attribute 2-5
API issues
 dictionary functions 3-10
 error codes 1-2
 open port 3-11
 ping 3-13
audio issues
 comparison tools 4-18
 embedded tags state 2-5
 gold master test 4-18
 packet sizes 3-11
 W3C SSML standard 2-4
 W3C SSML support 3-9
audio SSML tag 3-9

B

break SSML tag 2-4

C

cardinal say-as attribute 2-5
configuration issues
 W3C SSML standard 2-4

D

desc SSML tag 2-4
dictionary issues
 API functions 3-10
 error logging 3-10
 locations 3-13
 syntax 3-12
 XML 3-12
digits say-as attribute 2-5

E

embedded tags issues
 state 2-5

I

installation issues
 licensing 1-2

operating system 2-3
path 4-18
server 1-1
US English voice 2-3
voice packages 1-1

L

letters say-as attribute 2-5
lexicon SSML tag 2-4
logging issues
 dictionary errors 3-10
 event log scripts 2-6
 failures 3-14
 operating system log 4-17
 scripts 2-7
 SWItts_cbError 3-14
 SWIttsMessagePacket 3-14

M

metadata SSML tag 2-4

N

number say-as attribute 2-5

O

ordinal say-as attribute 2-5

P

prosody SSML tag 2-4

S

SAPI issues
 languages 2-6
say-as SSML tag 2-4, 2-5
 acronym 2-5
 cardinal 2-5
 digits 2-5
 letters 2-5
 number 2-5
 ordinal 2-5
 telephone 2-5
 words 2-5
script issues
 diagnostic logs 2-7
 error logs 2-7
 event logs 2-6
 server startup 3-10
 Windows services 2-6
server issues
 multi-threaded 4-17
 watch-dogging 4-17
SSML issues
 <audio> support 3-9

- validation [3-11](#)
- W3C SSML standard [2-4](#)
- STRT log event [2-6](#)
- sub SSML tag [2-4](#)
- SWItts_ALREADY_INITIALIZED [1-2](#)
- SWItts_DICTIONARY_ACTIVE [1-2](#)
- SWItts_DICTIONARY_INVALID_PRIORITY
[1-2](#)
- SWItts_DICTIONARY_INVALID_TYPE [1-2](#)
- SWItts_DICTIONARY_LOADED [1-2](#)
- SWItts_DICTIONARY_NOT_LOADED [1-2](#)
- SWItts_DICTIONARY_PARSE_ERROR [1-2](#)
- SWItts_DICTIONARY_PRIORITY_
ALREADY_EXISTS [1-2](#)
- SWItts_INVALID_MEDIATYPE [1-2](#)
- SWItts_LICENSE_ALLOCATED [1-2](#)
- SWItts_LICENSE_FREED [1-2](#)
- SWItts_NO_LICENSE [1-2](#)
- SWItts_SSML_PARSE_ERROR [3-11](#)
- SWItts_UNSUPPORTED [1-2](#)
- SWItts_URI_FETCH_ERROR [1-2](#)
- SWItts_URI_NOT_FOUND [1-2](#)
- SWItts_URI_TIMEOUT [1-2](#)
- SWIttsAddDictionaryEntry() [3-10](#)
- SWIttsDeleteDictionaryEntry() [3-10](#)
- SWIttsDictionaryLoad() [3-13](#)
- SWIttsGetDictionaryKeys() [3-10](#)
- SWIttsLookupDictionaryEntry() [3-10](#)
- SWIttsMigrateDictToXML [3-12](#)
- SWIttsOpenPort() [3-11](#)
- SWIttsOpenPortEx() [3-11](#)
- SWIttsPing() [3-13](#)
- SWIttsResetDictionary() [3-10](#)
- SWIttsSpeak() [3-11](#)

T

- telephone say-as attribute [2-5](#)
- tts.engine.dictionaries [3-13](#)
- tts.log.diagnostic.toSystemLog [4-17](#)
- tts.ssml.doStrictValidation [2-4](#), [3-11](#)

V

- voice issues
 - languages [2-6](#)
 - US English voice [2-3](#)
- voice SSML tag [2-4](#)

W

- W3C SSML tags [2-4](#)
- Windows issues
 - services [2-6](#)
- Windows service [2-6](#)
- words say-as attribute [2-5](#)