Migration Guide

SpeechWorks® solutions from ScanSoft®

Migrating to OpenSpeech™ Recognizer (OSR 3.0.9)

(for users of OSR 2.0, SpeechPearlXML, and previous releases of OSR 3.0)

Document History

Date	Release Name
November 2005	First Edition, Update 3 (for OSR 3.0.9)
December 2004	First Edition, Update 2 (for OSR 3.0.3)
August 2004	First Edition, Update (for OSR 3.0.1)
May 2004	First Edition (for OSR 3.0)

Notice

Published by:

ScanSoft, Inc.
Worldwide Headquarters
One Wayside Road
Burlington, MA 01803-4609
United States

Table of Contents

Chapter 1

Migrating to OSR 3.0

This document describes topics for platform and application developers who migrate existing OSR 2.0 or SpeechPearlXML 1.1 systems to the OSR 3.0 release.

Overview

OSR 3.0 adds new features and simplifies existing features without requiring code changes to OSR 2.0 systems. However, there are required and recommended migration tasks (which are summarized below).

New and changed information in First Edition, Update 3

- OSR 3.0.9 can be installed on Windows XP Professional. This operating system was not allowed previously. As a result, when migrating a SpeechPearlXML installation that runs on Windows XP, it is not necessary to change operating systems.
- OSR 3.0.9 adds support for the completetimeout configuration parameter. Previously, the parameter was ignored. See "New parameter support (completetimeout)" on page 23.
- Installations that use server-configured redundancy for their license servers should ensure that the machines operating in the quorum are running the same operating system. See "Check client/server installations" on page 13.

- A new special key called SWI_attributes is available in recognition results as of OSR 3.0.4. Applications are not required to use the key. See page 23.

- A new flag is added to SWIrecGetXMLResult( ) for users who want confidence scores provides as floating point numbers (as specified in the MRCP v2.06 specification). See “New recognition result support for MRCPv2” on page 24.

- A new <meta> tag is available to application developers who use robust parsing grammars. See “New robust parsing options” on page 26.

- A new format is allowed for speech grammars. The n-gram format is for advance grammar developers, and is based on a W3C draft proposal. See “New support for n-gram grammars” on page 22.

- OSR 3.0.9 allows speech grammars to import other grammars regardless of whether the grammars contain parser components. Previously, all grammars involved in an import needed to either have, or not have, parsers. See “Support for importing grammars” on page 22.

- There are several small changes for users of SpeakFreely grammars. The documentation is re-organized and expanded; there is a new tagging tool (tag_sf) and a compounding tool (compound_sf); the word compounding parameters are deprecated; and there are several new parameters. For links to the specific changes, see the preface of the *OSR Grammar Developer's Guide*.

- A new output format is provided for getting raw recognition results with SWIrecGetXMLResult( ). The wordlattice media type returns a format that can be used for semantic analysis of the recognized utterance. Unless you want to use the new format, there are no required migration issues. See the *OSR Reference Manual* for details.

- A new configuration parameter (swirec_max_source_grammar_size) allows you to limit the size of grammars that can be dynamically compiled. Unless you want to use the new parameter, there are no required migration issues. See the *OSR Reference Manual* for details.

- A new optimization level (11) is added to the grammar compiler (sgc). This is primarily used for applications with custom SLMs (Statistical Language Models). For details, see the *OSR Grammar Developer's Guide*.

Audience notes

Users of OSR 2.0 and earlier

This guide is primarily for application and platform developers who are migrating existing OSR 2.0 systems.

If you are migrating an earlier (OSR 1.*n*) system, you must use the *OSR 2.0 Migration Guide* to update your application or platform before migrating to OSR 3.0. (You do not need to install OSR 2.0 or run the application on that version, but you must make the changes described in the 2.0 migration guide.)

Important: some optional tasks in the *OSR 2.0 Migration Guide* become required when migrating to OSR 3.0. See “Required tasks for OSR 1.n systems” on page 6 for details.

New OSR users

If you are a platform developer who is integrating OSR into your platform for the first time, your primary resources should be the *OSR Platform Integration Manual* and the *OSR Reference Manual*. There is one topic in this migration guide that is not covered in those manual, see “Advanced endpointer” on page 21. Otherwise, you do not need this migration guide.

If you are an application developer who is building your first OSR application, your primary resource should be the application development documentation provided for your platform (not a ScanSoft product) and the *OSR Grammar Developer’s Guide*.

Users of previous releases of OSR 3.0

If you are a platform developer who has already migrated to OSR 3.0.0, 3.0.1, or 3.0.2, there is only one new topic in this updated migration guide: see “Advanced endpointer” on page 21.

Users of the SpeechPearlXML 1.1 recognizer

This guide includes a chapter for migrating SpeechPearlXML applications or platforms. See Chapter 6.

Contact ScanSoft technical support for assistance migrating your application.

Users of the SpeechWorks 6.5 recognizer

This guide does not address the needs of SpeechWorks 6.5 users. Contact ScanSoft technical support for assistance migrating your application.

How to use this guide

This guide has several layers of organization to help different users find the information they need. For example, below in this preface (page 5), is a summary of tasks for platform and application developers.

The chapters are organized as follows:

- Required tasks – issues that *must be addressed*. See *page 7*.
- Possibly required tasks – issues that *might need to be addressed*. See *page 8*.
- Recommended tasks – issues for review and consideration. See *page 14*.
- General issues – issues that typically require no action. See *page 21*.

The index of this guide groups issues by category. For example, “Audio issues,” “Configuration issues,” “Dictionary issues,” “Installation issues,” and so on.

New features and technical details

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

The OSR 3.0 product documentation contains hyperlinks to every topic discussed in this migration guide. See the “New and changed information” in the preface of each book:

- *OSR Client/Server Operator’s Guide*
- *OSR Grammar Developer’s Guide*
- *ScanSoft Licensing Handbook*
- *OSR Platform Integration Manual*
- *OSR Reference Manual*
- Language supplements for the supported languages

Platform tasks versus Application Development Tasks

Each migration task described in this guide indicates whether it should be performed by a platform developer, a system operator, or an application developer. To use the guide, review each task individually and determine whether it is appropriate for you.

Tasks for platform developers (or system operators)

Required migration tasks

Tasks:

- “Remove previous OSR versions” on page 7
- “Install OSR 3.0 and at least one language” on page 7

Required migration tasks (in some instances)

Tasks:

- “Discontinue use of old media types” on page 12
- “Check client/server installations” on page 13

Recommended migration tasks

Tasks:

- Review the “Tasks to support new features” on page 14

Tasks for application developers

Required migration tasks

Tasks:

- “Recompile binary grammars” on page 7

Required migration tasks (in some instances)

Tasks:

- “Migrate Asian-language pronunciation dictionaries” on page 8
- “Declare symbol sets in pronunciation dictionaries” on page 9
- “Convert old-style (OSR 1.n) dictionaries” on page 9
- “Update event log scripts” on page 10
- “Tune acoustic models” on page 11
- “Remove disallowed characters from grammars” on page 11

Recommended migration tasks

Tasks:

- “Change pronunciations from ARPAbet to SAMPA” on page 15
- “Update configuration parameters” on page 17
- “Evaluate voice enrollment applications” on page 20

Required tasks for OSR 1.*n* systems

This guide assumes you will migrate any OSR 1.*n* applications and platforms to OSR 2.0 before migrating them to OSR 3.0. It is not necessary to perform two separate migrations, but you must ensure that you have performed the tasks described in the *OSR 2.0 Migration Guide.*

In addition, some of the tasks described as optional in the *OSR 2.0 Migration Guide* are required for OSR 3.0.

- You must recompile all binary grammars in OSR 3.0 (see page 7). For OSR 2.0, it was only necessary to recompile large grammars.

- You must update all dictionaries to the XML format (page 9). For OSR 2.0, this was a recommendation.

- To load an OSR 2.0 grammar that conforms to the January 2001 specification into OSR 3.0, you must use the <meta> element. To load any grammars updated to the most recent SRGS specification (see page 21), you must use the <lexicon> element.

CHAPTER 2

Required migration tasks

Remove previous OSR versions

OSR 3.0 cannot be installed on a system where a previous OSR 2.0 is already installed. OSR 2.0 must be un-installed first. For instructions, see the “Installation Guide.txt” file in your OSR 2.0 baseline.

Install OSR 3.0 and at least one language

To have a functioning system, you must install the core OSR 3.0 software plus at least one language.

Recompile binary grammars

OSR 3.0 **does not accept** binary grammars that were compiled under OSR 2.0 or 1.*n*. All grammars must be re-compiled.

Compilation might fail if your grammar contains disallowed characters. See page 11 for details.

CHAPTER 3

Required in some instances

Many of the following tasks are required for migration depending on which OSR features are used on your system.

Migrate Asian-language pronunciation dictionaries

When creating pronunciation dictionaries for the following languages, you must use the SAMPA symbol set to define the pronunciations.

Languages that must convert to SAMPA
Cantonese (cn-HK)
Japanese (ja-JP)
Mandarin (zh-TW)

Extra details: In OSR 2.0, these languages used various sets of symbols to define pronunciations. For example, zh-TW used pinyin. These OSR 2.0 symbol sets are not supported in 3.0, and they must be converted to SAMPA.

Other OSR 2.0 languages (those not shown above) used the ARPAbet symbol set for pronunciations, and they can continue to use ARPAbet (perhaps temporarily until migrating dictionaries to SAMPA, see page 15). It is strongly recommended that all OSR 2.0 dictionaries be migrated as soon as possible to use SAMPA.

Declare symbol sets in pronunciation dictionaries

Except for the asian languages above, OSR 3.0 allows two symbol sets for pronunciations: SAMPA (the new set) and ARPAbet (supported for backwards compatibility).

For compatibility with OSR 2.0, OSR 3.0 assumes that dictionaries use ARPAbet symbols by default. But when you update (or create new) dictionaries with SAMPA, you must declare the symbol set with the “alphabet” attribute. The following example defines SAMPA for the entire dictionary; it also uses ARPAbet as a local override for the definition of tomato:

```
<?xml version="1.0" encoding="UTF-8" ?>
<lexicon xml:lang="en-us"
         alphabet="application/sampa;localization=swi">
   <entry key="George">
      <definition value="dZQdZ" />
   </entry>
   <entry key="tomato"
          alphabet="application/arpa;localization=swi" >
      <definition value="t ax m ey t ow" />
      <definition value="t aa m aa t ow" />
   </entry>
   <entry key="abc">
      <definition value="eI#sil#bi:#sil#si:" />
   </entry>
</lexicon>
```

The allowed values for the alphabet attribute are:

```
application/sampa;localization=swi
application/arpa;localization=swi     (this is the default)
```

Note: OSR 3.0 requires XML-formatted dictionaries. It does not support the non-XML format that was allowed in OSR 2.0 and 1.*n*. For details, see page 9.

Convert old-style (OSR 1.*n*) dictionaries

OSR 3.0 supports XML pronunciation dictionaries only. It does not support the dictionary format used in OSR 1.*n*. All old-style dictionaries must be converted to XML.

OSR 3.0 provides a command-line tool called conv_userdict to perform the conversion. See Appendix A.

Update event log scripts

This section describes changes to events, tokens, and token values in the OSR event log. If you have scripts that parse OSR event logs, and the scripts rely on specific event and token names, you might need to update the scripts. For details on the event log, see the *OSR 2.0 Reference Manual.*

Removed tokens

The OSR 3.0 event log has removed a couple tokens that were used in OSR 2.0. These tokens are removed from the SWIrcnd and SWIrrcn events:

DACC	String	Filename of the statistics file (the diphone accumulator) that tuned the acoustic model used for the recognition event. (Also, see the MACC token.)
NR	Boolean	Indicates whether the noisy speech filter was active for this utterance.

Added values for tokens

In OSR 3.0, the value of the MPNM token includes a <LangCode> indicator that was not present in OSR 2.0:

MPNM	Contains a comma separated list showing the language and acoustic model used to recognize the top choice on the n-best list. Each list element has the format <LangCode>/<RootModelName>.

In OSR 3.0.9, as part of the new support for the completetimeout configuration parameter, a value of “ctimeout” is added to the ENDR token of the SWIrcnd event. This value was not present in earlier versions of OSR.

Tune acoustic models

Some users of previous OSR releases have highly-customized acoustic models. These old models cannot be used with OSR 3.0.

Your options are:

- Use the new default 3.0 models.
- Tune the 3.0 models in the same way as done with the previous OSR models.

Install 3.0 and compare the accuracy of the default 3.0 models with previous accuracy tests on the customized models. If the accuracy of the new models is satisfactory, there is no need for further action.

Remove disallowed characters from grammars

Vocabulary items in OSR 2.0 grammars might contain characters that cause compilation failures in OSR 3.0. This situation is rare.

As described in the *OSR Grammar Developer's Guide*, the best characters for the text of vocabulary items in your speech grammars are:

Character Name	Character	Description
Digits	0-9	Individual digits, but not strings of digits.
Letters	a-z	Includes all alphabetic characters of the target language.

For example, it is better to fully spell the text of vocabulary items in your grammars. Compare the following:

```
<item> 50% </item>
<item> fifty percent</item>
```

In the example above, the second item is preferred because it explicitly defines what words can be spoken, and it avoids using non-alphabetic symbols (the percent sign) and strings of digits.

In OSR 2.0, the use of symbols and strings of digits in vocabulary items were discouraged, but usage was allowed in some languages. In OSR 3.0, the accepted usage has changed on a language-by-language basis. This affects application migration because OSR 3.0 will fail compilations of OSR 2.0 grammars if it cannot translate any contained symbols or strings of digits.

If a compilation fails, and you identify the cause as a non-alphabetic character or a string of digits, consider the following:

- You can edit the grammar to spell words instead of using the symbol or digit string.
- You can add pronunciations for symbols and digit strings into your user dictionaries. Changes to the grammar are still likely, but the grammar will be able to use the symbols and digits without spelling their names.
- You can instruct OSR to ignore items that have no pronunciation. Use the swirec_enable_robust_compile parameter as described in the *OSR Reference Manual*.

Discontinue use of old media types

This task is for platform integrators working on systems that originally supported OSR 1.n.

In OSR 1.n, the following media types were available for describing lost and suppressed audio samples in the SWIepAudioSamples and SWIrecAudioSamples structures:

- audio/x-suppressed
- audio/x-lost

These types were deprecated in OSR 2.0, but still allowed. (This is described in the *OSR 2.0 Migration Guide*.) The types are no longer allowed in OSR 3.0, and will generate an error if used. If you are using these types, you must change them to allowed values as described under the SWIrecAudioWrite( ) function in the *OSR Reference Manual*.

Check client/server installations

This task is for platform integrators (or system operators) with OSR installations that use server-configured redundancy for their license servers. (This is not a new installation requirement; it is a clarification that was omitted from the *ScanSoft Licensing Handbook*.)

Please ensure that the primary, secondary, and tertiary license servers forming the quorum are running the same operating system. Otherwise, OSR might fail to reconnect to license servers that return after being removed from service.

CHAPTER 4

Recommended migration tasks

Tasks to support new features

This section is for platform integrators. We strongly recommend that OSR 3.0 integrations support the features described in this section.

The following features were added in OSR 2.0, and might already be supported by your integration. However, because 2.0 was fully backward compatible, your current (migrations from 1.n to 2.0) might not support them and should be updated. For details on these integration tasks, see the *OSR Migration Guide* for OSR 2.0.

- Full Unicode (UTF-16) input into the API – OSR 3.0 supports unicode. To take advantage of this feature, the integrator must allow applications to supply all parameters and dynamic grammars in UTF-16. The platform must handle UTF-16 recognition results and pass them back to the application.

- SWIrecGrammarLoad( ) – Use the "uri/2.0" and "string/2.0" types to load grammars instead of the previous types of "uri" and "string". This is usually required for UTF-16 support, and also enables full SRGS compliance with respect to grammar media types.

  Also, the swirec_language VXIMap argument should be used with the SWIrecGrammarLoad( ) function. This is required for proper functioning of multi-lingual applications when they depend on built-in grammars and use the VoiceXML language property to control the active language. Strict VoiceXML 2.0 applications will likely require this feature.

- Magic-word and selective barge-in – The OSR 3.0 endpointer offers new barge-in modes. For example, these features support recognition of wake-up words in long streams of caller speech. To allow applications to use the features, you must support them in your endpointer integration. Your integration should include the use of the new SWIepRead( ) function. For details, see the *OSR Platform Integration Manual* and the *OSR Reference Manual*.

- ◻ OSR Server selection via API – OSR 3.0 allows the explicit selection of a recognition server; this feature helps you to develop resource management strategies for OSR Servers. To support the feature, your platform must expose the new SWIrecResourceAllocate( ) and SWIrecResourceFree( ) functions to the application (so that applications can select servers by resource group). See the *OSR Reference Manual* for information on the API functions.

- ◻ SWIrecGetXMLResult( ) – Use this new function instead of the older SWIrecGetKeyValue( ). This will simplify VoiceXML compliance. See the *OSR Reference Manual*.

- ◻ Parameter grammars – Platform integrators should allow application developers to load custom grammar types. Advanced application developers (speech grammar writers) will likely desire these types. For details, see the *OSR Grammar Developer's Guide*. For example, the following new types should be available:
  - application/x-swi-parameter – This type is needed for parameter grammars.
  - application/x-swi-wordlist – This type is needed for wordlist grammars.

- ◻ New error codes in the OSR API – The integration should be modified to handle these.

Change pronunciations from ARPAbet to SAMPA

All OSR 2.0 (and previous) pronunciation dictionaries used the ARPAbet symbol set to describe pronunciations. The ARPAbet symbols continue to work in OSR 3.0, but internally they are converted to the SAMPA symbol set. We recommend that you convert dictionaries as soon as possible.

The recommended way to convert dictionaries is to use the conversion tool described on page 49 to perform an initial migration and then check the results. The goal of your check is to reveal and improve any sub-optimal pronunciations resulting from the automated process. You can manually check the pronunciations by opening the converted dictionary file in an editor, or you can run accuracy tests (if you have this capability on your system).

Another way to convert dictionaries is to manually change pronunciations from ARPAbet to SAMPA. However, when you do this, you must declare the alphabet in the dictionary. See “Declare symbol sets in pronunciation dictionaries” on page 9.

For information on pronunciation symbols for any language, see the appropriate *OSR Language Supplement*.

Update configuration parameters

New parameters

New parameters are available for platform integrators to control the endpointer:

swiep_suppress_barge_in_time
swiep_suppress_waveform_logging
swiep_waveform_logging_channel_timeout
swiep_waveform_logging_max_channels

For details on these parameters, see the *OSR Reference Manual*.

Changes to grammar parameters

When tuning speech grammars, configuration parameters are used to refine performance and accuracy. Due to changes in the OSR 3.0 recognizer, you must re-tune any refinements in your 2.0 grammars that you want to retain. However, this re-tuning might not be necessary because OSR 3.0 is likely to improve accuracy of the grammars without the old refinements.

The following OSR 2.0 parameters are advanced features that were occasionally used for tuning. They are not supported in OSR 3.0, and they will be ignored. We recommend that you remove them (see Extra details below).

OSR 2.0 parameter	OSR 3.0 equivalent (with different values)
swirec_across_word_pruning	swirec_state_beam
swirec_fast_match_thresh	swirec_phoneme_lookahead_beam
swirec_lmscale	swirec_lmweight
swirec_max_forward_paths	swirec_max_arcs
swirec_search_threshold	none

To re-tune grammars, *you could map OSR 2.0 parameters* to the OSR 3.0 equivalents shown above. (The values are likely to be different.) However, there is no requirement to tune with the new parameters; in most cases, the default settings are sufficient. Before making changes, test grammars with the default values (the defaults are already defined in the baseline configuration), and then change values on a per grammar basis.

For details on configuration parameters, see the *OSR Reference Manual*.

Extra details

The following list should help you find instances of the OSR 2.0 parameters in your application:

swirec_across_word_pruning – could be defined in a user configuration file (user.xml); in source grammars using a <meta> tag; or as a runtime argument to the SWIrecRecognizerSetParameter( ) function.

swirec_fast_match_thresh – could be defined in a user configuration file (user.xml); in source grammars using a <meta> tag; or as a runtime argument to the SWIrecRecognizerSetParameter( ) function.

swirec_lmscale – could be defined in a user configuration file (user.xml) or in source grammars using a <meta> tag.

swirec_max_forward_paths – could be defined in a user configuration file (user.xml); in source grammars using a <meta> tag; or as a runtime argument to the SWIrecRecognizerSetParameter( ) function.

swirec_search_threshold – could be defined in a user configuration file (user.xml); in source grammars using a <meta> tag; or as a runtime argument to the SWIrecRecognizerSetParameter( ) function.

For example, do the following:

1. Search for occurrences of each parameter in your source code, user configuration file, and source grammar file.
2. For each parameter found, consider removing it (so that the default value will be used) or re-evaluate the parameter's value. Consult the *OSR Reference Manual* for the most recent information on the parameter and its setting.

Deprecated (removed) parameters

The following parameters were available (but not documented) in OSR 2.0. They are not available in OSR 3.0. If you use any of these parameters in OSR 3.0, the parameter is ignored and a warning message is written to the diagnostic log:

Deprecated parameters
swirec_acoustic_adapt_adapt_seg_model
swirec_acoustic_adapt_min_num_seg_utts
swirec_across_word_pruning (use swirec_state_beam instead)
swirec_add_to_thresh
swirec_allow_partial_results

Deprecated parameters
swirec_bound_config_name
swirec_boundary_weight
swirec_constraint_name
swirec_deletions_name
swirec_diphone_dimensions
swirec_disable_learn_dictionary
swirec_duration_weight
swirec_e_fsm_name
swirec_fast_match_thresh (use swirec_phoneme_lookahead_beam instead)
swirec_gramloc_across_word_pruning
swirec_language_dyn_lib_name
swirec_lmceil
swirec_lmfloor
swirec_lmoffset
swirec_log_full_grammar
swirec_log_inline_grammar
swirec_max_forward_paths (use swirec_max_arcs instead)
swirec_model_exp_scale
swirec_nword_info_name
swirec_nword_nnet_name
swirec_odig_weight
swirec_pc_fsm_name
swirec_pcs_name
swirec_realign_info_name
swirec_search_threshold
swirec_segmentation
swirec_short_list_mixtures
swirec_stw
swirec_system_lang_info_name
swirec_use_app_specific_learn_dictionary
swirec_vtln_info_name
swirec_vtln_level
swirec_wtw (use swirec_word_penalty instead)

Evaluate voice enrollment applications

If you have OSR 2.0 applications using the voice enrollment API (SWIve), we recommend that you re-enroll all utterances from your callers and re-create the associated pronunciation dictionaries. At a minimum, you should monitor the performance (recognition accuracy) of existing 2.0 enrollments with OSR 3.0.

Considering the following:

- With the OSR 3.0 recognizer, new enrollments will get better accuracy than re-used enrollments.
- You can continue to use dictionaries created from OSR 2.0 enrollments. However, the recognition accuracy of re-used enrollments is likely to be lower with the OSR 3.0 engine.

For voice enrollment concepts and programming overviews, see the *OSR Platform Integration Manual*. For API reference information, see the *OSR Reference Manual*.

CHAPTER 5

Miscellaneous migration issues

Advanced endpointer

OSR 3.0.3 supports an optional speech detector (known as the advanced endpointer) for use by platform integrators. The endpointer is responsible for distinguishing the end of speech in a spoken utterance.

To enable the advanced endpointer, see the instructions in the release notes.

The following parameters are not available when you use the new endpointer:

Deprecated parameters
swiep_prompt_speech_men_dur
swiep_prompt_speech_threshold
swiep_quiet_speech_min_dur
swiep_quiet_speech_threshold

Supported W3C grammar specification

OSR 3.0.9 supports the newest version of the Speech Recognition Grammar Specification Version 1.0. It is the W3C Proposed Recommendation December 18, 2003. For details, see http://www.w3.org/TR/speech-grammar/.

OSR 2.0 supported the previous Candidate Recommendation June 26, 2002.

The newest specification is fully compatible with the previous. Essentially, there are no differences between the two specifications.

New support for n-gram grammars

OSR 3.0.9 supports a new format for n-gram grammars. This feature is available to advanced grammar developers, and is described in the *OSR Grammar Developer's Guide*.

Use the following new media type when loading and activating the new grammar format:

application/x-swi-ngram+xml

Support for importing grammars

This is a topic for advanced speech grammar developers.

OSR 3.0.9 allows a speech grammar to import other grammars even if they do not have a parser component. Previously, there was a restriction all grammars involved in an import must either all have or all not have a parser component. This restriction primarily affected internal ScanSoft audiences and did not our customers.

Details:

- If a grammar containing no parser is imported by or imports a grammar that has a parser, then a dummy loop parser will be created for the grammar with no parser before importing. The dummy parser is a one-of loop of all the vocabulary including rule references to imported grammars. Once the dummy parser is generated, importing will proceed as normal, generating a composite grammar with a parser.
- The parser that is automatically generated does not have semantic scripting beyond the propagation of SWI_literal and SWI_spoken. Grammars with a parser that are imported by a grammar with none cannot pass semantic results; only SWI_literal is be available. Any imported grammars with no parser will

return their results in SWI_literal and their root rule (used for accessing results) will be named "root". For example:

```
<ruleref uri="slm_with_no_parser.gram"/>
<tag>SLMResult = "SLM: " + root.SWI_literal </tag>
```

New parameter support (completetimeout)

OSR 3.0.9 has added support for the completetimeout configuration parameter. Previously, the parameter was ignored.

Applications can use the parameter to fine-tune the end-of speech timer (controlled with the incompletetimout parameter). The two parameters are similar, but not identical: the completetimout timer is set to lesser values and it starts counting as soon as a valid recognition hypothesis is found. Thus, the new support provides a way to shorten delays after end-of-speech is detected.

See the *OSR Reference Manual* for complete details and cautions.

New special key (SWI_attributes)

As of OSR 3.0.4, a new special key called SWI_attributes is available (when keys are enabled using swirec_extra_nbest_keys). The key is available in the recognition result returned by the SWIrecGetXMLResult( ) function. Applications are not required to use the key.

The purpose of this key is to provide the attributes as character data to ensure that slot confidence scores are not lost. SWIrecGetXMLResult( ) returns an XML-format recognition result which contains slot confidences. These slot confidences are expressed as XML attributes. VoiceXML platforms typically transform the recognition result into an ECMAScript object that can be used by a voice application. However, there is no general standard that defines how platforms should make this transformation, and some platforms omit some information contained in the XML attributes.

Because advanced natural language applications rely on slot confidences, the SWI_ attributes key ensures that this information remains available in the ECMAScript objects returned by your browser platform. By representing slot confidence as character data in the XML format recognition result, the information should survive any platform transformations.

For example of recognition results containing this key, see the *OSR Grammar Developer's Guide*.

New recognition result support for MRCPv2

A new, optional flag is available for SpeechWorks Media Server (SWMS) users who want confidence scores provides as floating point numbers (as specified in the MRCP v2.06 specification). With the flag, the scores are returned as numbers between 0 and 1 (with up to 2 decimal values).

The flag is added to the emma media type format of the SWIrecGetXMLResult( ) function as follows:

```
application/x-vnd.speechworks.emma+xml;mrcpv=2.06
```

Removed parameters might generate warnings

OSR 3.0 has deprecated (removed) a number of configuration parameters that were present in OSR 2.0. These parameters were rarely used (and in most cases were not documented); most migrating applications will not notice their absence. See page 18 for details.

Raw scores have changed

Along with confidence scores, the recognizer can return intermediate results known as raw recognition scores. The range and magnitude of these scores has changed in OSR 3.0.

If your application grammars use the SWI_scoreDelta feature to influence raw scores, you do not need to change the assigned scoreDelta weights. (The weights are scaled internally to compensate for the change in raw scores.)

It is uncommon for applications to use raw scores in their calculations, but if you do you should consult with ScanSoft about the changes.

Update weights for alphanum and digits grammars

The alphanum and digit built-in grammars allow you to provide a list of constraints (a text file containing all valid strings). In addition, you can specify a weight for each entry. If your OSR 2.0 applications uses this feature, the weights can remain unchanged but their behavior might be different with the OSR 3.0 recognizer. You might need to re-tune the weights (typically by reducing their values) during your OSR 3.0 migration.

Extra information

Weights in constraining lists for these grammars are described in the *OSR Language Supplement* for each language. As a review of that information, the following VoiceXML line calls the alphanum grammar and specifies constraints listed in a file called const.txt:

```
builtin:grammar/alphanum?entries=const.txt
```

Inside const.txt, weights can be assigned to individual entries. For example:

```
::WEIGHTED 3
70zd2z -3
70zt2z 0
70dd2z 2.5
```

The value of WEIGHTED defines the maximum absolute value for the weight of any individual entry in the file. In this example, the range of adjustments is -3 to 3.

In OSR 3.0, lesser values for WEIGHTED should be used. The range of weight values should be more narrow, in the range of log probabilities.

For details on built-in grammars and entry lists, see the appropriate *OSR Language Supplement*.

New robust parsing options

A new <meta> tag (swirec_training_grammar) is allowed in robust parsing grammars, and a related parameter is allowed in user configuration files (swirec_max_training_grammar_size).

Previously in OSR 3.0, application developers were required to train SLMs (statistical language models) before compiling the robust parsing grammars that referred to the trained files. In OSR 3.0.9, the new tag swirec_training_grammar allows you to train the SLM during compilation.

Because the new tag causes the SLM to be trained every time the grammar is compiled, it is only recommended for small training sets. To avoid heavy performance costs, avoid using the tag when the grammar is compiled and loaded at runtime. The new configuration parameter swirec_max_training_grammar_size lets you limit the size of the training set.

Other robust parsing topics

For informational purposes, the following list describes additional robust parsing topics, which are documented in the *OSR Reference Manual* and the *OSR Grammar Developer's Guide*:

- swirec_compile_parser_with_weights – When you compile a robust parsing grammar, the compiler automatically adds weights to improve the recognition accuracy of concepts and fillers. To enable processing of these weights, the parameter swirec_compile_parser_with_weights is automatically be set to 1. This is done even if the robust parsing grammar contains a meta tag that sets this parameter to a different value. (In this case a warning message is written to the error log.) It is not possible to override this automatic setting.

- swirec_fsm_grammar – This parameter is used in robust parsing grammars. It specifies a finite state machine (fsm) used by a speech grammar. It was previously described in the *OSR Robust Parsing Supplement* (which is now removed from the OSR documentation set).

- swirec_fsm_wordlist – This parameter is used in robust parsing grammars. It specifies a wordlist used by a speech grammar. It was previously described in the *OSR Robust Parsing Supplement* (which is now removed from the OSR documentation set).
- swirec_max_parses_per_literal – This parameter is automatically set to one for robust parsing grammars (unless you override the automatic setting by specifying this parameter as a <meta> in the grammar file).
- swirec_max_training_grammar_size – This parameter limits the size of any Statistical Language Model (SLM) that is trained as part of grammar compilation (instead of being trained before compilation).
- swirec_nbest_list_length – For robust parsing grammars, this parameter is automatically set to a value that provides a good quality of slot confidence values. This can lead to increased CPU time. If necessary, you can set swirec_nbest_list_length to a different (lower) value in a meta tag in a robust parsing grammar. If you make such a change, you should carefully evaluate the effect upon recognition accuracy and the quality of slot confidence values.
- swirec_training_grammar – This parameter is used when training a Statistical Language Model (SLM). It refers to the SLM training set from within a grammar so that the SLM can be generated during grammar compilation.

CHAPTER 6

Migrating SpeechPearlXML to OSR 3.0

This chapter shows the main differences between SpeechPearlXML and OSR 3.0 and provides assistance to migrate from SpeechPearlXML to OSR 3.0. It does not provide detailed step by step instructions. Instead this guide should serve as a quick starting guide for SpeechPearlXML users.

Hardware and software requirements

The hardware and software requirements for both recognizers are similar, but not equal.

Approximate hardware requirements; see the installation guide for most recent details:

SpeechPearlXML	OSR 3.0
On Windows (SP XML 1.0, 1.1) ☐ Single processor Intel Pentium III 650 MHz or faster On Unix (SP XML 1.3) ☐ Sun SPARC processor (V8plusa architecture version)	☐ Intel Pentium III or Pentium 4-based computer (the processor must support streaming SIMD extensions)

Approximate software requirements; see the installation guide for most recent details:

SpeechPearlXML	OSR 3.0
Windows Software Requirements: ☐ One of the following: • MS Windows 2000 Professional (Service Pack 3) • MS Windows XP Professional SP1a • MS Windows 2000 Server ☐ Adobe Acrobat Reader 5 or later ☐ (Optional) Microsoft Visual C++ 6.0 Unix Software Requirements ☐ SunOS 5.8 (Solaris 8)	Windows Software Requirements: ☐ One of the following: • Windows 2000 Professional or Server Service Pack 2 • MS Windows XP Professional SP2 • MS Windows 2003 Server ☐ Adobe Acrobat Reader 4 or later ☐ (Optional) Microsoft Visual C++ 6.0 with Service Pack 3 Linux Software Requirements: ☐ Red Hat Linux 7.2 or Red Hat Enterprise Linux 3.0 ☐ Adobe Acrobat Reader 4 or later ☐ (Optional) An ANSI C compiler

System components

This section shows variations between the OSR and SpeechPearlXML components. Here is a general overview:

Feature	SpeechPearlXML	OSR 3.0
Available Architectures	☐ Standalone ☐ Client / Server	☐ Standalone (all-in-one) ☐ Client / Server
Default Installation Path	$SPPEARLDIR = <d>:\scansoft\telephonyasr\speechpearl	$SWISRSDK = <d>:\Program Files\SpeechWorks\OpenSpeech Recognizer
Samples	$SPPEARLDIR/spike	$SWISRSDK/samples/swirec_sample
Demonstration Language Resource	Demo language resource "en-demo" is automatically installed and used by the sample programs.	The swirec_sample requires the language resource "en-US" to be installed for a successful sample execution.
Voice activity Detection	SpeechDetector	Endpointer (automatically installed with the OSR core)

Feature	SpeechPearlXML	OSR 3.0
License Manager	FlexLM Version 9	FlexLM version 9

License manager

Although both products install Macrovision FlexLM License Manager version 9, there are important distinctions as shown in the table below. The license key of SpeechPearlXML cannot be used with OSR 3.0. Contact your ScanSoft Sales Representative if you have questions regarding the terms of your license agreement.

Component	SpeechPearlXML	OSR 3.0
License manager version	Version 9	Version 9
License file	License.dat	osr.lic
Default location	C:\scansoft\telephonyasr\lmgr\system	$SWISRSDK\flexlm\license folder
Log File	...\lmgr\log.txt	...flexlm\license folder\osr-lic.log
Lmtools	...\lmgr\bin	...flexlm\components
Configuration	LM_LICENSE_FILE (Environment Variable)	SWILicenseServerList (Configuration Parameter)
Configuration Settings	C:\scansoft\telephonyasr\lmgr\system\license.dat	port@hostname (default 27000@localhost) If OSR is configured to check out licenses from several license server, then other license servers are added separated by semicolon on Windows. For other operating systems, see the *ScanSoft Licensing Handbook*. port@hostname1;port@hostname2
How to get a license key?	Via Orderdesks. United States: request.support@scansoft.com Rest of World: orderdesk.eu@scansoft.com	Via the License Fulfillment website: http://licensing.speechworks.com
Related Document	Spxml_installation.pdf	OSR Licensing Handbook.pdf

Voice activity detector

Both products provide voice activity detection. The main differences between the SpeechPearlXML Speech Detector and the OSR 3.0 endpointer is that the OSR endpointer is license protected and that end-of-utterance detection is done by the OSR recognizer and not by the OSR endpointer.

Language resources

There are many similarities between OSR and SpeechPearlXML language resources. In the table below, “lg” and “co” are abbreviations for <Language> and <Country>. For example: en-US is English in the United States, and en-GB is English in Great Britain.

SpeechPearlXML	OSR 3.0	Description
Installation Directory $SPLANGDIR/lg-co	Installation Directory $SWISRSDK/config/lg.co	For OSR, the language package is installed to the same target directory as the recognizer. You cannot change the installation directory as you could do it for SpeechPearl.
Background Lexicon \lg-co\standard\lex\ background\lg-co.bld	System Dictionary $SWISRSDK/config/lg.co/ dictionary/Basic.dictionary	System dictionaries are based on the SpeechPearl background lexicon and have the OSR format with SAMPA phoneme set. Only one system dictionary per language can be loaded with OSR.
User Lexicon uiuser.lex	User Dictionary name.userdict	OSR only supports the XML format. OSR 3.0 provides a conversion tool (page 49) to covert SpeechPearlXML user dictionaries.
Statistical Auto-Transcription Resource lg-co.lrf	Automatic Dictionary lg-co.lrf	OSR’s Automatic Dictionary is based on the SpeechPearlXML Statistical Auto-Transcription Resource.

SpeechPearlXML	OSR 3.0	Description
Precedence of Dictionaries 1. User Lexicon 2. Background Lexicons 3. Auto-Transcription Resource	Precedence of Dictionaries 1. User Dictionary 2. System Dictionary 3. Automatically generated pronunciations	The default precedence of the dictionaries is the same for OSR and SpeechPearl. The difference is that the default precedence can be changed for OSR by defining the user dictionary as a "backup" dictionary with the result that it has equal precedence with the system dictionary and OSR will use all pronunciations from both dictionaries.
Acoustic Models lg-co.amo lg-co_accopt.amo lg-co_runopt.amo	Acoustic Models *.fsm (finite state machine) *.hmm (hidden markov model)	OSR delivers a single acoustic model consisting of two files.
Configuration File SpeechPearl configuration file	Configuration File lg-co.xml	Each OSR language comes with a language specific XML parameter file.
Built-in Grammars According to the VoiceXML 2.0 specification. Installation Dir: $SPLANGDIR/lg-co/ standard/builtins_si	Built-in Grammars According to the VoiceXML 1.0 specification. Installation Dir: $SWISRSDK/config/lg.co/contexts	OSR built-in grammars are provided as binary grammars. The list of grammars varies for each language.

Migration Tasks

1. “Remove installed SpeechPearl version”
2. “Install OSR 3.0”
3. “Update project settings”
4. “Update header files”
5. “Migrate from SpeechPearl API to OSR API”
6. “Migrate from SpeechDetector API to endpointer API”
7. “Migrate the configuration”
8. “Update endpointer configuration”
9. “Migrate SpeechPearlXML user lexicons”
10. “Migrate SpeechPearlXML grammars”

Remove installed SpeechPearl version

We strongly recommend that you remove SpeechPearlXML and all of it’s sub-components (including the License Manager) before installing OSR 3.0.

For instructions, see the SpeechPearlXML "spxml_installation.pdf" file.

Install OSR 3.0

To have a functioning system, you must install the core OSR 3.0 software plus at least one language.

For instructions, see the OSR 3.0 Installation Guide (Installation Guide.txt).

Update project settings

When migrating from a SpeechPearlXML Standalone application to an OSR All-In-One application, the following changes have to be made to the Project Settings:

Project Settings	SpeechPearlXML	OSR 3.0
C/C++ Preprocessor Additional Include Directories	$(SPPEARLDIR), $(SPPEARLDIR)\spike\utils, $(SPPEARLDIR)\..\SpeechDetector	$(SWISRSDK)\include, $(SWICOMMONSDK)\include
Link objects	$(SPPEARLDIR)\lib\dsspxmlapi01r.lib $(SPPEARLDIR)\..\SpeechDetector\lib\dsvdq02r.lib	SWIrec.lib SWIep.lib

Update header files

Update the header files to be included for OSR 3.0 as follows:

SpeechPearlXML	OSR 3.0
// SpeechPearlXML interface #include "ds/spxmlapi.h"	// OSR Recognizer #include "SWIrecAPI.h"
// SpeechDetector API #include "ds/vadq.h"	// OSR endpointer #include "SWIepAPI.h"
// SpeechPearlXML types #include "ds/spdefs.h"	
// SpeechPearlXML environment variables #include "ds/environ.h"	
// SpeechPearlXML C wrapper functions #include "ds/cwrap.h"	
// SpeechDetector C wrapper functions #include "ds/cwrap_sd.h"	

Migrate from SpeechPearl API to OSR API

The following table shows the API functions of SpeechPearlXML and OSR 3.0 to provide assistance on the migration from an existing SpeechPearlXML integration to OSR 3.0. Please note, that not all of the functions can be mapped one-to-one. Thus the mapping mainly concentrates on the functionality provided by the API functions and not on the input / output parameters or status. The functions are presented similarly to the order of their appearance:

SpeechPearlXML	OSR 3.0	Description
DS_SP_getVersion( )	SWIrecGetParameter( ) with parameter swirec_version SWIepGetParameter( ) with parameter swiep_version	Returns the software version information.
DS_SP_getTraceFlag( )		Retrieves the currently set trace options.
DS_SP_setTraceFlag( )		Enables or disables tracing.
DS_SP_getErrorText( )		Returns the textual representation for a given return code value.
None	SWIrecLogEvent( )	This function writes an event and related information into the recognizer event log. It can be called at any time.
DS_SP_create( )	SWIrecInit( )	System (Process) Initialization based on configuration files.
DS_SP_createFromKTV( )		System Initialization based on the settings given in the KTV parameters.
DS_SP_getSystemParam( )		Retrieves a system-specific parameter.

SpeechPearlXML	OSR 3.0	Description
DS_SP_open( )	SWIrecRecognizerCreate( )	All-In-One: Creates a recognizer resource. OSR Client/Server: Establishes a connection between C/S and allocates a recognizer resource on the server. Licensing: Allocates a license if the license mode is set to default.
DS_SP_requestResources( )	SWIrecResourceAllocate( )	Client/Server: Has to be called to allocate a recognizer resource on the server create if the server selection mode is set to explicit. Licensing: Needs to be called to allocate a license if the licensing mode is set to explicit.
DS_SP_registerOnEvent( )		Registers a user-provided notification callback function for an engine.
DS_SP_getEngineParam( )	SWIrecRecognizerGetParameter( )	Retrieves the values of recognition parameters.
	SWIrecRecognizerSetParameter( )	Sets the value of a recognition configuration parameter.
	SWIrecAcousticStateReset( )	Resets the acoustic state, because OSR adapts to the caller and stores the data. The reset needs to be done at the beginning of each call.
	SWIrecAcousticStateLoad( )	Is used when multiple recognizer resources are used for a single call. Loads an acoustic state object from a memory buffer to preserve adaptation that the recognizer has already performed on the caller.

SpeechPearlXML	OSR 3.0	Description
	SWIrecAcousticStateQuerySize( )	Determines the buffer size to allocate for the input to SWIrecAcousticStateSave( ).
	SWIrecAcousticStateSave( )	Writes an acoustic state object to a memory buffer to allow acoustic state information to be transferred across recognizers.
	SWIrecRecognizerSetSessionName( )	Platform should set channels and application names for logging.
DS_SP_loadGrammarFromUri( )	SWIrecGrammarLoad( )	OSR: Fetches a grammar from URI or string (inline) and loads it into memory. Automatically compiles the grammar. SP XML: Loads a grammar from URI.
DS_SP_loadGrammarFromUriEx( )	SWIrecGrammarLoad( )	SP XML: Loads top-level built-in grammars.
DS_SP_loadInlineGrammar( )	SWIrecGrammarLoad( )	SP XML: Loads an inline grammar.
	SWIrecGrammarFree( )	Signals the recognizer that the grammar is no longer needed. The grammar is removed from cache if the recognizer needs free space.
	SWIrecGrammarCompile( )	Is used for pre-compiling at runtime. OSR client/server: Not supported.
DS_SP_activateConcepts( )	SWIrecGrammarActivate( )	SP XML: Activates a set of concepts. OSR: Activates a loaded grammar. Automatically loads and compiles the grammar.

SpeechPearlXML	OSR 3.0	Description
	SWIrecGrammarDeactivate( )	Deactivates grammar for the next recognitions. It remains in the cache.
DS_SP_putDTMFData( )	SWIrecParseDTMFResults( )	Parses a DTMF string against the currently active grammars.
DS_SP_putAudioData( )	SWIrecRecognizerStart( )	Starts recognition using currently active grammars.
DS_SP_putAudioData( )	SWIrecAudioWrite( )	Deliver audio samples to the recognizer.
	SWIrecRecognizerCompute( )	Compute results for current recognition. (end of utterance detection is done by the recognizer).
DS_SP_putEndOfSentence( )		Signals the end of an utterance and forces the completion of the recognition and grammar processing.
DS_SP_retrieveResultBufferSize( )		Computes the buffer size needed for retrieving the XML recognition result.
DS_SP_retrieveResult( )	SWIrecGetXMLResult( )	Retrieve an XML representation of recognition results.
	SWIrecGetWaveform( )	Retrieves the waveform for the last recognition utterance in µ-law format.
DS_SP_retrieveWordGraphBufferSize( )		Provides the buffer size required for retrieving the SpeechPearlXML recognition result in word graph format.
DS_SP_retrieveWordGraph( )		Provides the SpeechPearlXML recognition result in word graph format.

SpeechPearlXML	OSR 3.0	Description
DS_SP_putWordGraph( )		Feeds the speech understanding component of SpeechPearlXML with a word graph.
DS_SP_destroyResult( )		Destroys the internal SpeechPearlXML result object.
DS_SP_stop( )	SWIrecRecognizerStop( )	Abort current recognition.
DS_SP_releaseResources( )	SWIrecResourceFree( )	OSR C/S: Frees an assigned OSR server when this server session was created via SWIrecResourceAllocate( ). Licensing: Frees a license for the specified recognizer resource that has been allocated via SWIrecResourceAllocate( ).
DS_SP_releaseResources( )	SWIrecRecognizerDestroy( )	OSR client-server: Releases the connection to the server and frees server resources if the server selection mode is set to explicit.
DS_SP_close( )	SWIrecRecognizerDestroy( )	OSR all-in-one: Destroys a recognizer resource. Licensing: Releases the license allocated by SWIrecRecognizerCreate( ) if the licensing mode is set to default.
DS_SP_destroy( )	SWIrecTerminate( )	System shutdown.
	SWIrecThreadCleanup( )	Clean up memory allocation for thread. This function is for OSR client-server environments only, and only for integrations that use transient threads to make SWIrecAPI calls.

See the *OSR Platform Integration Manual* and the *OSR Reference Manual* for more information on the recognizer API.

Migrate from SpeechDetector API to endpointer API

The table below shows the API functions of the SpeechPearlXML SpeechDetector and the OSR 3.0 endpointer to provide assistance on the migration from an existing SpeechPearlXML application to OSR 3.0. Please note, that not all of the functions can be mapped one-to-one from SpeechDetector to endpointer. Thus the following mapping table mainly concentrates on the functionality provided by the API functions and not on the input / output parameters or status. The functions are presented similarly to the order of their appearance:

SpeechDetector 2.1 SpeechPearlXML	Endpointer OSR	Description
DS_VADQ_create( )	SWIepInit( )	Speech detector initialization based on configuration files. Call this function after SWIrecInit.
DS_VADQ_ getChannelParam( )	SWIepGetParameter( )	Get the value for one of the speech detector parameters.
DS_VADQ_ setChannelParam( )	SWIepSetParameter( )	Set the value for one of the speech detector parameters. Call SWIepSetParameter( ) before SWIepStart( ). Parameter values are valid for the duration of the next utterance only.
DS_VADQ_open( )	SWIepDetectorCreate( )	Creates a speech detector resource (channel). OSR: Each speech detector resource requires a license. SWIepDetectorCreate checks out a license on the license server.
DS_VADQ_reset( )	SWIepAcousticStateReset( )	The speech detector adapts to the acoustic state of the caller and calling environment during a call. The platform must call SWIepAcousticStateReset( ) either at the beginning of a new telephone call or at the end of a call in order to reset acoustic state information for a new caller. Do not call this function during a single call.
None	SWIepAcousticStateLoad( )	Reads a previously saved acoustic state information to exchange the acoustic state for one call that is processed by speech detector resources on multiple machines. Acoustic state for the speech detector and the recognizer are not interchangeable since they contain different data.
None	SWIepAcousticStateQuerySize( )	Determines the buffer size to allocate for the input to SWIepAcousticStateSave( ).

SpeechDetector 2.1 SpeechPearlXML	Endpointer OSR	Description
	SWIepAcousticStateSave( )	Writes an acoustic state object to a memory buffer to allow acoustic state information to be transferred across speech detectors.
	SWIepPromptDone( )	The speech detector uses different thresholds to detect speech depending on whether the prompt is playing. While a prompt is playing, the caller typically must speak slightly louder - to trigger the speech detector - than after the prompt is done.
	SWIepResourceAllocate( )	The swiep_licensing_mode parameter must be set to explicit for SWIepResourceAllocate( ) to work. Determines if there are available licenses on the license server. If so, SWIepResourceAllocate( ) locks a license.
DS_VADQ_start( )	SWIepStart( )	Initialize speech detector for a new detection.
DS_VADQ_analyze( )	SWIepWrite( )	Deliver audio samples to the speech detector.
DS_VADQ_getAudioData( )	SWIepRead( )	Retrieves audio data containing speech from the speech detector.
DS_VADQ_stop( )	SWIepStop( )	Indicate that the current utterance is complete and stop detection.
DS_VADQ_setTraceFlag( )	None	Activates the trace function of the SpeechDetector.
DS_VADQ_getTraceFlag( )		Retrieves the current setting of the trace flag.
DS_VADQ_getVersion( )		Returns the speech detector version information.
	SWIepResourceFree( )	Explicitly frees the user license for the specified endpointer resource.
DS_VADQ_close( )	SWIepDetectorDestroy( )	Destroys a speech detector resource (channel, state). OSR: The license is checked in on the license server.
DS_VADQ_destroy( )	SWIepTerminate( )	Shuts down the speech detector. Call this function after SWIrecTerminate( ).

See the *OSR Platform Integration Manual* and the *OSR Reference Manual* for further information on the endpointer API, operating modes, and configuration.

Migrate the configuration

This section describes the configuration of the OSR 3.0 All-In-One configuration.

As OSR comes with a more flexible configuration mechanism than SpeechPearlXML, the following table provides an overview into OSR configuration:

OSR 3.0 Configuration	Description
Baseline.xml	This file is provided during OSR installation and provides the default settings. It also defines values for a "default" language.
<lg-co>.xml	Every language has a language-specific baseline configuration file to override the "default" language settings of the baseline.
User.xml	Overrides the Baseline.xml. Settings in this file can override the settings for the default language (found in the Baseline.xml configuration file) as well as any other language (found in a language-specific configuration file).
SpeechWorks.cfg	This configuration file contains static parameters, like licensing and logging parameters, after the initial installation.
VoiceXML	Some parameters can be set in VoiceXML scripts. The platform passes the settings to OSR with the SWIepSetParameter( ) and SWIrecRecognizerSetParameter( ) functions.
Grammar files	Many parameters can be controlled with the <meta> tag inside XML grammars. The settings are passed to OSR when the you activate the grammar for recognition.
API	Many parameters can be set only with the SWIepSetParameter( ) and SWIrecRecognizerSetParameter( ) functions. These parameters are dynamic in the sense that the values might change from one recognition event to the next.
Parameter grammar files	Any parameter that can be set via SWIrecRecognizerSetParameter( ) can also be set in a parameter grammar file. These grammars are simple text files that list recognition parameters.
OSR	Some parameters are read-only. They are set by OSR components.

For further information on the configuration mechanism and the order of precedence, see the *OSR Reference Manual.*

The following sections show those SpeechPearlXML standalone configuration parameters that have an equivalent OSR parameters. Other parameters are not listed here.

As OSR provides many more configuration parameters than those shown in the following sections, see the *OSR Reference Manual* for complete details.

License management configuration

In contrast to SpeechPearlXML, OSR licensing monitors the recognizer engines and the endpointer engines. OSR provides one feature for the recognizer and one for the endpointer. The vocabulary size is not controlled by OSR licensing. Licensing can be set to explicit for both, the OSR recognizer and the OSR endpointer:

SpeechPearlXML	OSR	Description
lateLicenseCheckout	swirec_licensing_mode	Sets the licensing mode for the recognizer. Set this OSR parameter in the "default" language section of a user configuration file to change Baseline.xml default.
	swiep_licensing_mode	Sets the licensing mode for the endpointer. Set this OSR parameter in the "default" language section of a user configuration file to change Baseline.xml default.
lmFeatureName		SpeechPearlXML feature name to be checked out by SpeechPearl on the license server.
lmUserData		Path to the SpeechPearlXML license file.

Tracing and logging parameters

The OSR logging mechanism differs from the SpeechPearlXML logging in the following way: OSR differentiates between TRC and Event logging and creates two separate log files.

TRC logging

OSR writes diagnostic and error logging through a mechanism called TRC. The filenames, locations, and other characteristics of the logs are controlled by configuration variables. TRC log files are encoded as UTF-8 by default.

TRC defines tags which represent some family, class, or subsystem within an application. Logging related operations are internally driven by whether a given tag has been enabled. TRC diagnostic messages are configured in simple text files called tagmap files. TRC parameters can be set via the SpeechWorks.cfg configuration file or environment variables.

The following table shows those SpeechPearlXML logging parameters that can be mapped to OSR 3.0 TRC logging parameters:

SpeechPearlXML	OSR 3.0	Description
spTraceFlag	DiagTagMapsBaseline	This parameter points to the specific tagmap files used for TRC logging. The default is: $SWISRSDK/config/defaultTagmap.txt; $SWISRSDK/config/bwcompatTagmap.txt
logDir	DiagFileName	This parameter defines the default TRC output logging file. Default is $SWISRSDK/trc.log.
aliasmap.txt	DiagErrorMap	This parameter specifies a language-specific error definition file, where error codes are given names and canonical messages. For example: $SWISRSDK/config/SWIErrors.en.us.txt
numRollingNMSLogMessages	DiagMaxFileSizeKB	Sets the maximum size of the rolling TRC log file.
	DiagConfigPollSecs	Defines how often the tagmap files are reloaded to avoid to restart applications when enabling and disabling tags for TRC diagnostic logging.

Event logging

OSR writes an event log containing recognition statistics and results during execution. In addition, the recognizer can log audio files (waveforms) containing endpointed spoken utterances. OSR controls the filenames of event logs, but other aspects of the logs are user defined. By default, event logs are encoded as UTF-8

Event logging parameters can be set via the SpeechWorks.cfg configuration file or environment variables.

The following table introduces often used OSR 3.0 event logging parameters together with their mapping SpeechPearlXML parameters and their meaning.

SpeechPearlXML	OSR	Description
	DataCaptureDirectory	Determines the location where event logs are written.

SpeechPearlXML	OSR	Description
	DataCaptureDiskSpace	Defines the maximum size of event log and waveform data written. Set this parameter in the SpeechWorks.cfg configuration file. Alternatively, set it as an environment variable.
	DataCaptureRolloverTime	This parameter sets the time for a daily rollover of the event log.
logAudioData	swirec_suppress_waveform_logging	This parameters controls waveform logging to the event log.

Recognition parameters

Below are the SpeechPearlXML recognition parameters that have an equivalent OSR recognition parameter. When switching from the SpeechPearlXML SR parameters to the OSR recognizer parameters, you cannot use the parameter settings of SpeechPearlXML. Please use their default settings as provided by the OSR installation instead.

SpeechPearlXML	OSR	Description
nBest	swirec_nbest_list_length	Maximum number of n-best answers that can be returned.
audioFormat	swirec_audio_media_type	Defines the formats of the audio that will be supplied to the recognizer.
SRAcousticTolerance	swirec_state_beam	This parameter is the main parameter for guiding the Viterbi beam search. It is used to prune all active search paths. Use the OSR default settings when migrating from SpeechPearlXML to OSR 3.0.
SRSyntacticTolerance	swirec_word_beam	The word beam applies only at word endings. The hypothesis is that search paths at end of words can be pruned tighter than other paths, without loss of accuracy. Use the OSR default settings when migrating from SpeechPearlXML to OSR 3.0. This parameter is not documented in the OSR documentation set.
SRWordPenalty	swirec_word_penalty	Word transition penalty parameter. Use the OSR default settings when migrating from SpeechPearlXML to OSR 3.0. This parameter is not documented in the OSR documentation set.

Grammar loading parameters

Migrate the following parameter to load a grammar from URI:

SpeechPearlXML	OSR	Description
httpProxyName	swirec_inet_proxy_server	Names an HTTP proxy server to be used by OSR when fetching grammar URIs.
	swirec_inet_proxy_server_port	Names the port of an HTTP proxy server. This parameter is required if swirec_inet_proxy_ server is defined.

Language definition parameters

Each OSR language installs a language-specific baseline configuration file, like en-us.xml with language resource en-US. This file contains language specific settings equal to the SpeechPearlXML language definition parameters and overrides the "default" language settings of the baseline.xml file.

SpeechPearlXML	OSR	Description
languageNName		Language Name
languageNDefaultAmo	swirec_global_fsm_name swirec_model_name	Acoustic Model
languageNUserLex	swirec_user_dict_name	User Dictionary
languageNBGLex	swirec_system_dict_name	System Dictionary
languageNStatisticalTS	swirec_statistical_ts_name	Automatic Dictionary

Update endpointer configuration

The OSR baseline configuration file also contains the endpointer configuration parameters. Some of them are equal to the SpeechDetector configuration parameters. To override these parameters, please make use of the configuration mechanism as described above.

SP XML SpeechDetector	OSR Endpointer & Recognizer	Description
vadTraceFlag	DiagTagMapsBaseline	This parameter points to the tagmap files used for TRC logging. The default is: $SWISRSDK/config/defaultTagmap.txt; $SWISRSDK/config/bwcompatTagmap.txt
maxUtteranceDuration	maxspeechtimeout	Defines the maximum duration (in milliseconds) of an utterance after the begin-of-speech point in the audio stream.
maxEndSilenceDuration	incompletetimeout	Controls the length of a period of silence after callers have spoken to conclude that they finished. This timer is also known as the "end of speech" timeout.
loudnessInitialThreshold	sensitivity	Controls sensitivity of the speech detector when looking for speech.
temporaryThreshold	swiep_in_prompt_sensitivity_percent	Controls how loudly the caller must speak to interrupt prompts (barge-in) in order for speech to be detected.
DS_VADQ_open	swiep_audio_media_type	Informs the speech detector about the audio formats that will be supplied to it.
	swiep_mode	Sets the endpointer into various modes.

Migrate SpeechPearlXML user lexicons

Use the dictionary conversion tool (conv_userdict) to migrate your SpeechPearlXML lexicons to the OSR 3.0 user dictionaries. See Appendix A for a description of the tool.

Migrate SpeechPearlXML grammars

The OSR XML grammar syntax is based on the same specification as the SpeechPearlXML grammar which is the "Speech Recognition Grammar Specification for the W3C Speech Interface Framework: http://www.w3.org/TR/speech-grammar". Therefore no syntax updates are required to XML grammars when migrating from SpeechPearlXML to OSR 3.0.

As far as semantic interpretation is concerned, OSR 3.0 does not support semantic interpretation for speech recognition according to the W3C working draft (2003-04-01) (http://www.w3.org/TR/semantic-interpretation/ as SpeechPearlXML does. Therefore please follow these steps to migrate your SpeechPearlXML grammar to an OSR 3.0 XML grammar:

	SpeechPearlXML	OSR 3.0
Update tag format of the grammar element	from tag-format="ssft-semantics/1.0"	to tag-format="swi-semantics/1.0"
Migrate semantic interpretation	from SSFT semantics	to the OSR semantic interpretation described in the *Grammar Developer's Guide*

APPENDIX A

Conversions

Converting OSR 1.*n* and OSR 2.0 dictionaries to OSR 3.0

The dictionary conversion tool (conv_userdict) automates the process of migrating OSR 1.*n* and OSR 2.0 pronunciation dictionaries to OSR 3.0. However, automated conversions are not perfect and still require manual checking and tuning of the results.

The primary function of the conversion tool is to substitute the old-style ARPAbet pronunciations with the new-style SAMPA definitions. In addition, it will automatically convert the OSR 1.*n* text format to the required XML format. The tool is stored in the bin directory of your OSR 3.0 installation.

Usage

```
conv_userdict -i <input> -o <output> -l <langcode>
  -oa <alphabet>
```

Parameter	Description
input	Required input. Filename of input user dictionary.
output	Output. Filename of converted xml user dictionary. If omitted, output is sent to stdout.
langcode	Input. The default language of the dictionary. For example, en-US or de-DE. This parameter is required unless the language is explicitly declared in the dictionary file.
-oa	Optional input. determines the pronunciation symbol set of the output dictionary: "sampa" writes SAMPA to the output dictionary, and "none" writes the same alphabet as found in the input file. The default is “none”.

Notes

When you run this tool, all messages are printed to stderr.

The tool automatically detects the format of the input dictionary. The following formats are supported:

- OSR 1.*n* user dictionaries in text format
  (encoding ISO-8859-1, ARPAbet pronunciations)
- OSR 2.0 user dictionaries in XML format
  (encoding ISO-8859-1, ARPAbet pronunciations)
- SpeechPearl user dictionaries (encoding UTF-8, SAMPA pronunciations)

The output format is always the OSR 3.0 XML user dictionary format (UTF-8 encoding).

You can use the "-i" option more than once. This will merge several user dictionaries into one new dictionary. In this case, the tool will merge all pronunciations for each word.

All words coming from SpeechPearl dictionaries are changed from upper to lowercase.

The conversion tool uses a language-dependent file to map ARPAbet characters to SAMPA. To verify the conversions being performed (or to change the mapping), see the appropriate mapping file in the OSR baseline:

```
config/<LangCode>/dictionary/<LangCode>_arpa2sampa.txt
```

SpeechPearl configuration conversions

For systems migrating from the SpeechPearl recognition engine to OSR, the following approximations should help speed the conversion process:

Pearl Parameter	OSR 3.0 Parameter	Conversion Formula
statePruningOffset	swirec_state_beam	swirec_state_beam = ((statePruningOffset / 10000) - 15) * 7

Pearl Parameter	OSR 3.0 Parameter	Conversion Formula
wordPruningOffset	swirec_word_beam	swirec_word_beam = ((wordPruningOffset / 10000) - 15) * 7
phonemeLAPruningOffset	swirec_phoneme_lookahead_ beam	swirec_phoneme_lookahead_beam = ((phonemeLAPruningOffset / 10000) - 15) * 7
silenceOffset	swirec_silence_pruning_offset	swirec_silence_prune_offset = (silenceOffset / 10000) * 7
maxNumSearchNodes	swirec_max_arcs	swirec_max_arcs = maxNumSearchNodes
modLoopPenalty	swirec_loop_penalty	swirec_loop_penalty = modLoopPenalty / 10000
modNextPenalty	swirec_next_penalty	swirec_next_penalty = modNextPenalty / 10000
modSkipPenalty	swirec_skip_penalty	swirec_skip_penalty = modSkipPenalty / 10000
modPausePenalty	swirec_pause_penalty	swirec_pause_penalty = modPausePenalty / 10000
modSilencePenalty	swirec_silence_penalty	swirec_silence_penalty = modSilencePenalty / 10000
wholeWordUnitPenalty	swirec_whole_word_unit_ penalty	swirec_whole_word_unit_penalty = wholeWordUnitPenalty / 10000
wordPenalty	swirec_word_penalty	swirec_word_penalty = wordPenalty / 10000

Other conversion tools

OSR provides tools for VoiceXML browser developers or platform integrators who are migrating applications that ran on SpeechWorks 6.5 (or earlier) or OSR 1.n. Specifically, the tools can be used for the following:

- Vocabulary and grammar files (.avf and .bnf) from SpeechWorks 6.5 formats to the VoiceXML formats required by the OpenSpeech Recognizer (OSR).

- Grammar files (.xml) from OSR 1.n to OSR 2.0.

For information on these tools, see the *OSR 2.0 Migration Guide*.

Index