UniMRCP

PocketSphinx Plugin

Usage Guide

Revision: 3

Created: February 16, 2017

Last updated: May 20, 2017

Author: Arsen Chaloyan

Table of Contents

1 Overview

This guide describes how to configure and use the PocketSphinx plugin to the UniMRCP server. The document is intended for users having a certain knowledge of PocketSphinx and UniMRCP.

1.1 Installation

For installation instructions, use one of the guides below.

- RPM Package Installation (Red Hat / Cent OS)
- Deb Package Installation (Debian / Ubuntu)

1.2 Applicable Versions

Instructions provided in this guide are applicable to the following versions.

UniMRCP 1.4.0 and above UniMRCP PocketSphinx Plugin 1.0.0 and above

2 Configuration Format

The configuration file of the PocketSphinx plugin is located in */opt/unimrcp/conf/umspocketsphinx.xml* and the relevant data files are placed in the directory */opt/unimrcp/data/pocketsphinx*.
The configuration file is written in XML.

2.1 Document

The root element of the XML document must be *<umspocketsphinx>*.

Attributes

Name	Unit	Description
license-file	File path	Specifies the license file. File name may include patterns containing '*' sign. If multiple files match the pattern, the most recent one gets used.

Parent

None.

Children

Name	Unit	Description
<model-collection>	String	Specifies a collection of the PocketSphinx acoustic models.
< speech-dtmf-input-detector>	String	Specifies parameters of the speech and DTMF input detector.
< utterance-manager>	String	Specifies parameters of the utterance manager.

Example

This is an example of a bare document.

```
< umspocketsphinx license-file="umspocketsphinx_*.lic">
</ umspocketsphinx>
```

2.2 Model Collection

This element contains a collection of the PocketSphinx language/acoustic models.

Attributes

Name	Unit	Description
default-language	String	Specifies the default language to use, if not set by the client.

Parent

<umspocketsphinx>

Children

Name	Unit	Description
<model>	String	Specifies the language/acoustic model.

Example

This is an example of a bare collection of models.

```
<model-collection default-language="en-US">
</model-collection>
```

2.3 Model

This element specifies a PocketSphinx language/acoustic model.

Attributes

Name	Unit	Description
enable	Boolean	Specifies whether the model is enabled or disabled.
language	String	Specifies a language the model is made for.
sampling-rate	Integer	Specifies a sampling rate the model is made for.

acoustic-model-dir	Dir path	Specifies a directory containing the acoustic model data.
dictionary	File path	Specifies a dictionary file to use.
grammar-dir	Dir path	Specifies a directory containing built-in speech grammars.

Parent

<model-collection>

Children

None.

Example

The example below defines two en-US language models: one is for audio sampled at 8 kHz, the other – for 16 kHz.

```
<model-collection default-language="en-US">
  <model
    enable="true"
    language="en-US"
    sampling-rate="8000"
    acoustic-model-dir="cmusphinx-en-us-8khz"
    dictionary="cmudict-en-us.dict"
    grammar-dir="speech-grammar-en-us"
  />
  <model
    enable="true"
    language="en-US"
    sampling-rate="16000"
    acoustic-model-dir="cmusphinx-en-us-16khz"
    dictionary="cmudict-en-us.dict"
    grammar-dir="speech-grammar-en-us"
  />
</model-collection>
```

2.4 Speech and DTMF Input Detector

This element specifies parameters of the speech and DTMF input detector.

Attributes

Name	Unit	Description

speech-start-timeout	Time interval [msec]	Specifies how long to wait in transition mode before triggering a start of speech input event.
speech-complete-timeout	Time interval [msec]	Specifies how long to wait in transition mode before triggering an end of speech input event.
noinput-timeout	Time interval [msec]	Specifies how long to wait before triggering a no-input event.
input-timeout	Time interval [msec]	Specifies how long to wait for input to complete.
dtmf-interdigit-timeout	Time interval [msec]	Specifies a DTMF inter-digit timeout.
dtmf-term-timeout	Time interval [msec]	Specifies a DTMF input termination timeout.
dtmf-term-char	Character	Specifies a DTMF input termination character.
normalize-input	Boolean	Specifies whether received spoken input stream should be normalized or not.
speech-leading-silence	Time interval [msec]	Specifies desired silence interval preceding spoken input. The parameter is used if normalize-input is set to true.
speech-trailing-silence	Time interval [msec]	Specifies desired silence interval following spoken input. The parameter is used if normalize-input is set to true.
speech-output-period	Time interval [msec]	Specifies an interval used to feed speech frames to the PocketSphinx recognizer.

Parent

<umspocketsphinx>

Children

None.

Example

The example below defines a typical speech and DTMF input detector having the default parameters set.

```
<speech-dtmf-input-detector
```

```
        speech-start-timeout="300"
        speech-complete-timeout="1000"
        noinput-timeout="5000"
        input-timeout="10000"
        dtmf-interdigit-timeout="5000"
        dtmf-term-timeout="10000"
        dtmf-term-char=""
        normalize-input="true"
        speech-leading-silence="300"
        speech-trailing-silence="300"
        speech-output-period="80"
      />
```

2.5 Utterance Manager

This element specifies parameters of the utterance manager.

Attributes

Name	Unit	Description
save-waveforms	Boolean	Specifies whether to save waveforms or not.
waveform-base-uri	String	Specifies the base URI used to compose an absolute waveform URI.
waveform-folder	Dir path	Specifies a folder the waveforms should be stored in.
expiration-time	Time interval [min]	Specifies a time interval after expiration of which waveforms are considered outdated.
purge-waveforms	Boolean	Specifies whether to delete outdated waveforms or not.
purge-interval	Time interval [min]	Specifies a time interval used to periodically check for outdated waveforms.

Parent

<umspocketsphinx>

Children

None.

Example

The example below defines a typical utterance manager having the default parameters set.

```
<utterance-manager
  save-waveforms="false"
  waveform-base-uri="http://localhost/utterances/"
  waveform-folder=""
  expiration-time="60"
  purge-waveforms="true"
  purge-interval="30"
/>
```

3 Configuration Steps

This section outlines common configuration steps.

3.1 Using Default Configuration

The default configuration and data files correspond to the en-US language and should be sufficient for the general use.

3.2 Specifying Models

While the default configuration and data files contain references to an en-US acoustic model and a dictionary file, which are getting installed with the package *unimrcp-pocketsphinx-model-en-us*, other acoustic models and dictionary files can also be used.

In order to add a new or modify the existing model, the following parameters must be specified.

- language the model is made for
- sampling rate the acoustic data corresponds to
- path to a directory containing acoustic model
- path to a dictionary file

Note that, unless an absolute path is specified, the path is relative to the directory */opt/unimrcp/data/pocketsphinx/$language*.

The following example defines two models: one for en-US and the other for de-DE language.

```
<model-collection default-language="en-US">
  <model
    enable="true"
    language="en-US"
    sampling-rate="8000"
    acoustic-model-dir="cmusphinx-en-us-8khz"
    dictionary="cmudict-en-us.dict"
    grammar-dir="speech-grammar-en-us"
  />
  <model
    enable="true"
    language="de-DE"
    sampling-rate="8000"
    acoustic-model-dir="cmusphinx-de-de-8khz"
    dictionary="cmudict-de-de.dict"
    grammar-dir="speech-grammar-de-de"
  />
```

```
        </model-collection>
```

3.3 Specifying Built-in Grammars

Built-in grammars are stored in the *fsg* format and can be referenced by the client by means of a built-in URI, such as:

```
builtin:speech/$name
```

where $name is the name of a grammar file stored in the specified speech grammar directory for a particular model.

For instance, the package *unimrcp-pocketsphinx-model-en-us* installs a sample grammar file called *command.fsg*, located in the directory */opt/unimrcp/data/pocketsphinx/en-US/speech-grammar-en-us*. This sample grammar can be referenced by the client using the following URI:

```
builtin:speech/command
```

3.4 Specifying Speech/DTMF Input Detector

The default parameters specified for the speech and DTMF input detector are sufficient for the general use. However, various timeouts can be adjusted to better suite a particular requirement.

- speech-start-timeout

  This parameter is used to trigger a start of speech input. The shorter is the timeout, the sooner a START-OF-INPUT event is delivered to the client. However, a short timeout may also lead to a false positive.

- speech-complete-timeout

  This parameter is used to trigger an end of speech input. The shorter is the timeout, the shorter is the response time. However, a short timeout may also lead to a false positive.

- noinput-timeout

  This parameter is used to trigger a NO-INPUT event. The parameter can be overridden per MRCP session by setting the header field NO-INPUT in SET-PARAMS and RECOGNIZE requests.

- input-timeout

  This parameter is used to limit input time. The parameter can be overridden per MRCP session by setting the header field RECOGNITION-TIMEOUT in SET-PARAMS and RECOGNIZE requests.

- dtmf-interdigit-timeout

This parameter is used to set inter-digit timeout on DTMF input. The parameter can be overridden per MRCP session by setting the header field INTER-DIGIT-TIMEOUT in SET-PARAMS and RECOGNIZE requests.

- dtmf-term-timeout

This parameter is used to set termination timeout on DTMF input and is in effect when dtmf-term-char is set and there is a match for an input grammar. The parameter can be overridden per MRCP session by setting the header field INTER-DIGIT-TIMEOUT in SET-PARAMS and RECOGNIZE requests.

- dtmf-term-char

This parameter is used to set a character terminating DTMF input. The parameter can be overridden per MRCP session by setting the header field INTER-DIGIT-TIMEOUT in SET-PARAMS and RECOGNIZE requests.

3.5 Specifying Utterance Manager

The default parameters specified for the speech and DTMF input detector are sufficient for the general use. However, various timeouts can be adjusted to better suite a particular requirement.

- save-waveforms

Utterances can optionally be recorded and stored if the configuration parameter save-waveforms is set to true. The parameter can be overridden per MRCP session by setting the header field SAVE-WAVEFORMS in SET-PARAMS and RECOGNIZE requests.

- waveform-base-uri

This parameter specifies the base URI used to compose an absolute waveform URI returned in the header field WAVEFORM-URI in response to RECOGNIZE requests.

- waveform-folder

This parameter specifies a path to the directory used to store waveforms in.

- expiration-time

This parameter specifies a time interval in minutes after expiration of which waveforms are considered outdated.

- purge-waveforms

This parameter specifies whether to delete outdated waveforms or not.

- purge-interval

This parameter specifies a time interval in minutes used to check for outdated waveforms if purge-waveforms is set to true.

4 Supported Features

4.1 Supported MRCP Methods

- ✓ DEFINE-GRAMMAR
- ✓ RECOGNIZE
- ✓ START-INPUT-TIMERS
- ✓ SET-PARAMS
- ✓ GET-PARAMS

4.2 Supported MRCP Events

- ✓ RECOGNITION-COMPLETE
- ✓ START-OF-INPUT

4.3 Supported MRCP Header Fields

- ✓ Input-Type
- ✓ No-Input-Timeout
- ✓ Recognition-Timeout
- ✓ Waveform-URI
- ✓ Media-Type
- ✓ Completion-Cause
- ✓ Start-Input-Timers
- ✓ DTMF-Interdigit-Timeout
- ✓ DTMF-Term-Timeout
- ✓ DTMF-Term-Char
- ✓ Save-Waveform
- ✓ Speech-Language
- ✓ Cancel-If-Queue

4.4 Supported Grammars

- ✓ JSGF
- ✓ Built-in/extendable FSG speech grammars

- ✓ Built-in/embedded DTMF grammar(s)

4.5 Supported Results

- ✓ NLSML

5 Usage Examples

5.1 JSGF Grammar

This examples demonstrates how to load a JSGF grammar using a DEFINE-GRAMMAR request and further reference the loaded grammar in a RECOGNIZE request.

C->S:

```
MRCP/2.0 603 DEFINE-GRAMMAR 1
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
Content-Type: application/x-jsgf
Content-Id: request1@form-level
Content-Length: 432

#JSGF V1.0;
grammar cmdGrammar;
public <basicCmd> = <startPolite> (<callCommand> | <dialCommand>) <endPolite>;

<callCommand> = call <name>;
<dialCommand> = dial <digit>;

<name> = (steve | young | bob | johnston | john | jordan | joe);
<digit> = (one | two | three | four | five | six | seven | eight | nine | zero | oh);

<startPolite> = (please | kindly | could you) *;
<endPolite> = [ please | thanks | thank you ];
```

S->C:

```
MRCP/2.0 112 1 200 COMPLETE
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
Completion-Cause: 000 success
```

C->S:

```
MRCP/2.0 305 RECOGNIZE 2
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
No-Input-Timeout: 5000
Recognition-Timeout: 10000
Start-Input-Timers: true
Confidence-Threshold: 0.87
```

```
Save-Waveform: true
Content-Length: 27

session:request1@form-level
```

S->C:

```
MRCP/2.0 83 2 200 IN-PROGRESS
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
```

S->C:

```
MRCP/2.0 115 START-OF-INPUT 2 IN-PROGRESS
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
Input-Type: speech
```

S->C:

```
MRCP/2.0 492 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: 5091ac2cc2f8284d@speechrecog
Completion-Cause: 000 success
Waveform-Uri: <http://localhost/utterances/utter-5091ac2cc2f8284d-
2.wav>;size=20480;duration=1280
Content-Type: application/x-nlsml
Content-Length: 208

<?xml version="1.0"?>
<result>
  <interpretation grammar="request1@form-level" confidence="1.00">
    <instance>call steve</instance>
    <input mode="speech">call steve</input>
  </interpretation>
</result>
```

5.2 Built-in Speech Grammar

This examples demonstrates how to reference a built-in speech grammar in a RECOGNIZE request. The built-in speech grammar *command* is defined in the *command.fsg* file located in the directory */opt/unimrcp/data/pocketsphinx/en-US/speech-grammar-en-us*.

C->S:

```
MRCP/2.0 333 RECOGNIZE 1
Channel-Identifier: 3a9672600dc64b61@speechrecog
Content-Id: request1@form-level
Content-Type: text/uri-list
Cancel-If-Queue: false
No-Input-Timeout: 5000
Recognition-Timeout: 10000
Start-Input-Timers: true
Confidence-Threshold: 0.87
Save-Waveform: true
Content-Length: 22

builtin:speech/command
```

S->C:

```
MRCP/2.0 83 1 200 IN-PROGRESS
Channel-Identifier: 3a9672600dc64b61@speechrecog
```

S->C:

```
MRCP/2.0 115 START-OF-INPUT 1 IN-PROGRESS
Channel-Identifier: 3a9672600dc64b61@speechrecog
Input-Type: speech
```

S->C:

```
MRCP/2.0 478 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier: 3a9672600dc64b61@speechrecog
Completion-Cause: 000 success
Waveform-Uri: <http://localhost/utterances/utter-3a9672600dc64b61-
1.wav>;size=25920;duration=1620
Content-Type: application/x-nlsml
Content-Length: 194

<?xml version="1.0"?>
<result>
  <interpretation grammar="command" confidence="1.00">
    <instance>dial five</instance>
    <input mode="speech">dial five</input>
  </interpretation>
</result>
```

5.3 Built-in DTMF Grammar

This examples demonstrates how to reference a built-in DTMF grammar in a RECOGNIZE request.

C->S:

```
MRCP/2.0 266 RECOGNIZE 1
Channel-Identifier: d26bef74091a174c@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Start-Input-Timers: true
Confidence-Threshold: 0.7
Speech-Language: en-US
Dtmf-Term-Char: #
Content-Length: 19

builtin:dtmf/digits
```

S->C:

```
MRCP/2.0 83 1 200 IN-PROGRESS
Channel-Identifier: d26bef74091a174c@speechrecog
```

S->C:

```
MRCP/2.0 113 START-OF-INPUT 1 IN-PROGRESS
Channel-Identifier: d26bef74091a174c@speechrecog
Input-Type: dtmf
```

S->C:

```
MRCP/2.0 382 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier: d26bef74091a174c@speechrecog
Completion-Cause: 000 success
Content-Type: application/x-nlsml
Content-Length: 197

<?xml version="1.0"?>
<result>
  <interpretation grammar="builtin:dtmf/digits" confidence="1.00">
```

```
  <input mode="dtmf">1 2 3 4</input>
  <instance>1234</instance>
 </interpretation>
</result>
```

5.4 Speech and DTMF Grammars

This examples demonstrates how to reference a built-in DTMF grammar and a speech grammar combined in a RECOGNIZE request. In this example, the user is expected to input a 4-digit pin.

C->S:

```
MRCP/2.0 327 DEFINE-GRAMMAR 1
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
Content-Type: application/x-jsgf
Content-Id: grammar-1
Content-Length: 166

#JSGF V1.0;
grammar pinCmd;
public <pin> = <digit> <digit> <digit> <digit>;
<digit> = ( one | two | three | four | five | six | seven | eight | nine );
```

S->C:

```
MRCP/2.0 112 1 200 COMPLETE
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
Completion-Cause: 000 success
```

C->S:

```
MRCP/2.0 275 RECOGNIZE 2
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Start-Input-Timers: true
Confidence-Threshold: 0.7
Speech-Language: en-US
Content-Length: 47

builtin:dtmf/digits?length=4
session:grammar-1
```

S->C:

```
MRCP/2.0 83 2 200 IN-PROGRESS
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
```

S->C:

```
MRCP/2.0 115 START-OF-INPUT 2 IN-PROGRESS
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
Input-Type: speech
```

S->C:

```
MRCP/2.0 399 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: 6ae0f23e1b1e3d42@speechrecog
Completion-Cause: 000 success
Content-Type: application/x-nlsml
Content-Length: 214

<?xml version="1.0"?>
<result>
  <interpretation grammar="grammar-1" confidence="1.00">
    <instance>one two three four</instance>
    <input mode="speech">one two three four</input>
  </interpretation>
</result>
```

6 References

6.1 CMUSphinx Tutorials

- Basic concepts of speech
- Overview of CMUSphinx toolkit
- Building the dictionary
- Building the language model
- Adapting existing acoustic model
- Building the acoustic model

6.2 Specifications

- Speech Recognizer Resource
- Java Speech Grammar Format
- NLSML Results