

## GRAMMAR Description File

```
*****  
mp34me.grammar  
*****
```

```
; Radio station band
```

```
BAND (  
[  
a {return("am")}  
f {return("fm")}  
]  
?m  
?band  
)
```

```
; single digits
```

```
DIGIT [  
[zero oh] {return("0")}  
one {return("1")}  
two {return ("2")}  
three {return ("3")}  
four {return ("4")}  
five {return ("5")}  
six {return ("6")}  
seven {return ("7")}  
eight {return ("8")}  
nine {return ("9")}  
]
```

```
;80-90 for FM
```

```
FM-TENS [  
eighty {return ("8")}  
ninety {return ("9")}  
]
```

```
;tens
```

```
TENS [  
ten {return("10")}  
eleven {return("11")}  
twelve {return("12")}  
thirteen {return("13")}  
fourteen {return("14")}  
fifteen {return("15")}  
sixteen {return("16")}  
twenty {return ("20")}  
thirty {return ("30")}  
forty {return ("40")}  
fifty {return ("50")}  
sixty {return ("60")}  
seventy {return ("70")}  
eighty {return ("80")}  
ninety {return ("90")}  
]
```

```
;decimal point
```

```

DEC [
[dot point] {return (".")}
]

; six eight zero
THREE-D (DIGIT:d1 DIGIT:d2 DIGIT:d3) {return(strcat($d1 strcat ($d2 $d3)))}

; one oh five zero
FOUR-D (DIGIT:d1 DIGIT:d2 DIGIT:d3 DIGIT:d4) {return(strcat($d1 strcat($d2
strcat($d3 $d4)))})}

; six eighty
THREE-SHRT (DIGIT:d1 TENS:d2) {return(strcat ($d1 $d2))}

; fourteen thirty
FOUR-SHRT (TENS:d1 TENS:d2) {return(strcat ($d1 $d2))}

; nine eight point one
TWO-D-ONE (DIGIT:d1 DIGIT:d2 DEC:d3 DIGIT:d4) {return(strcat($d1 strcat($d2
strcat($d3 $d4)))})}

; one oh four point five
THREE-D-ONE (DIGIT:d1 DIGIT:d2 DIGIT:d3 DEC:d4 DIGIT:d5) {return(strcat($d1
strcat($d2 strcat($d3 strcat($d4 $d5)))})}

; ninety eight point one
TWO-S-ONE (FM-TENS:d1 DIGIT:d2 DEC:d3 DIGIT:d4) {return(strcat($d1 strcat($d2
strcat($d3 $d4)))})}

; ninety point three
TENS-S-ONE (TENS:d1 DEC:d2 DIGIT:d3) {return(strcat($d1 strcat ($d2 $d3)))}

; check for valid station number, 87.3 < X < 107.9 or 530 < X < 1600?

STATION [
THREE-D:f FOUR-D:f THREE-SHRT:f FOUR-SHRT:f TWO-D-ONE:f THREE-D-ONE:f TWO-S-
ONE:f TENS-S-ONE:f
] {return ($f)}

;phrase definition

.Sentence [
(BAND:b STATION:f) {<station_band $b> <station_freq $f>}
(STATION:f BAND:b) {<station_band $b> <station_freq $f>}
]

*****
mp34me.slot-definitions
*****

station_band
station_freq

```

## Nuance – Crash Course

The Nuance voice-recognition software is capable of extracting recorded waveforms into natural language patterns. However, it requires the reference of a grammar list to accurately pigeonhole a spoken utterance into a useful range of interpreted words used by our application.

The grammar file is divided into ‘subgrammar’ definitions followed by the actual phrase definition, which is declared as `.sentence` located at the end of the `mp34me.grammar` file.

Examples of the subgrammar names are **BAND**, **TENS**, **FM-TENS**, **DEC**, **TWO-D-ONE**, **STATION**, and so on. Notice that subgrammars can contain other subgrammars, and that subgrammars can return values to their parent grammars. This hierarchal structure facilitates quick grammar changes and flexible grammar creation.

The filtered variables at the end of the grammar structure contain the station band in a character string and the station frequency in a floating-point number. These two values are contained by two slot definitions, `station_band` and `station_freq`, respectively. These two values are handed off to the next stage of the voice processing engine for validity checks and pushed to the next portion of our product architecture.

### Syntax Description

|                                                                          |                                                                                                                                                                                                                |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>[word1 word2]</code>                                               | word1 or word2 will be recognized and interpreted. For example, using <code>[oh zero]</code> to denote the different sayings of the 0 (zero) numeral.                                                          |
| <code>?m</code>                                                          | the word m is optional to be spoken. In our implementation, the a or f utterance is all that’s necessary to recognize the station band. Also, word band is also an optional word in the <b>BAND</b> subgrammar |
| <code>DIGIT</code><br><code>[one {return ("1")}</code><br><code>]</code> | The spoken word one will return the value “1” as a <b>DIGIT</b> subgrammar.                                                                                                                                    |
| <code>DIGIT:f</code>                                                     | The returned value from subgrammar <b>DIGIT</b> is stored in the variable f . From above, the variable will the returned value “1”.                                                                            |
| <code>Strcat(\$f \$d)</code>                                             | The two variables f and d are concatenated.                                                                                                                                                                    |