| Table of Contents |
|-------------------|------|
| Preface           | 5    |
| Audience          | 5    |
| Conventions       | 5    |
| References        | 6    |
| **1. Nuance Grammars** | 7    |
| GSL Basics        | 7    |
| Subgrammars       | 8    |
| Field Grammars    | 9    |
| Form Grammars     | 9    |
| Ambiguous Grammars| 10   |
| Independent Grammars and Rulesets | 10 |
| Referencing Grammars | 11 |
| Using an Inline Grammar | 11 |
| Referencing an External Grammar | 11 |
| Grammar Scopes    | 13   |
| Independent Grammars | 13 |
| Rule Sets         | 13   |
| Grammar Syntax    | 14   |
| Rule Names        | 14   |
| Grammar Expressions | 14 |
| Grammar Assignment Commands | 17 |
| Value Expressions | 17   |
## 2. XML Speech Grammars

### XML Grammar Basics

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>21</td>
</tr>
<tr>
<td>Inline and External Grammars</td>
<td>22</td>
</tr>
<tr>
<td>Default Rule</td>
<td>22</td>
</tr>
<tr>
<td>Language and Pronunciation</td>
<td>22</td>
</tr>
<tr>
<td>Tag Summary</td>
<td>22</td>
</tr>
</tbody>
</table>

- `<example>`: 23
- `<grammar>`: 24
- `<item>`: 25
- `<one-of>`: 26
- `<rule>`: 27
- `<ruleref>`: 28
- `<tag>`: 29
- `<token>`: 30
VoiceXML applications use grammars to specify sets of valid user utterances at particular points in an interaction with the application. For example, at the beginning of your application, you may ask the user to select among a set of predefined options. In your VoiceXML document, you'll use a grammar to identify the set of possible things a user can say for each option. The speech-recognition engine uses the grammar to identify which option the user is selecting.

A VoiceXML application references a grammar with the <grammar> element. The grammar can be either:

- An inline grammar description (included directly in your VoiceXML code).
- An external grammar described in a separate file.

BeVocal VoiceXML supports grammars specified in either of two forms:

- Nuance Grammar Specification Language (GSL)
- XML Speech Recognition Grammar Format

Audience

This document is for software developers using the BeVocal Café development environment. It assumes that you are familiar with the basic concepts of HTML and that you already have some familiarity with VoiceXML authoring.

Conventions

**Bold** font is used for:
- Headings

**Fixed width** font is used for:
- Code examples
- Tags and attributes
- Values or text that must be typed as shown

**Italic fixed width** font is used for:
- Variables
- Prototypes or templates; what you actually type will be similar in format, but not the exact same characters as shown

**Italic** font is used for:
- Introducing terms that will be used throughout the document
- Emphasis
References

For additional or related information, you can refer to:

- **VoiceXML Reference.** BeVocal.  
  (http://cafe.bevocal.com/docs/vxml/index.html)
  http://www.voicexml.org under Specs for PDF  
  (http://www.w3.org/TR/2000/NOTE-voicexml-20000505 for HTML)
- **Nuance Grammar Developer’s Guide.** Nuance.  
  Under developers documentation after logging on at the Nuance web site  
  (http://www.nuance.com).
This chapter describes the Nuance Grammar Specification Language (GSL).

GSL Basics

A grammar identifies different words or phrases that a user might say, and (optionally) specifies how to interpret a valid expression in terms of values for field item variables. Grammars can range from a simple list of possible words to a complex set of possible phrases.

Grammars for BeVocal VoiceXML applications can be written in the Nuance Grammar Specification Language (GSL).

A grammar in GSL consists of one or more rules.

Each rule in a grammar typically has three parts:

1. A rule name (optional), which must begin with a capital letter and identifies the rule for use in other rules.
2. A grammar expression (required), which defines the possible utterances associated with that rule.
3. Assignment commands (optional), which specify what values to assign to field item variables for each valid utterance.

You create grammar expressions using a set of grammar operators to combine words and/or other rules. To understand the examples in this section, you just need to know three of the grammar operators:

- **[]** – disjunction. For example, \([a \ b]\) means \(a\) or \(b\).
- **()** – sequence. For example, \((a \ b)\) means \(a\) followed by \(b\).
- **?** – optional. For example, \(?a\) means \(a\) is optional.

Since most grammars in VoiceXML identify a set of possible words that a user might say, the grammar description is usually enclosed in square brackets \([\]\) to represent disjunction.

Although assignment commands can follow each grammar expression, they are not always needed. For example, you don’t need them for grammars in <link> or <choice> elements. If the user speaks a phrase that matches an expression in a link grammar, the <link> element is simply activated; if assignment commands are present, they are ignored.

The following example shows a grammar rule without assignment commands.

```
SummerMonths [june july august]
```

Rules for grammars that appear within <field> or <form> elements generally have assignment commands. Assignment commands have the form:

```
{<slotname value>}
```
A *slot name* identifies the field item variable to be assigned the specified value when a user utterance matches the grammar expression.

The following example shows a grammar rule with assignment commands.

```ruby
PrimaryColors [
  ( [red pink burgundy] )
    { <color red> }
  ( [(?sky blue) turquoise] )
    { <color blue> }
  ( [(?forest green chartreuse] )
    { <color green> }
]
```

Assignment commands are associated with the grammar expression that immediately precedes them. Each time the speech-recognition engine encounters a word or phrase that matches an expression in an active grammar, it flags a match and executes any assignment command associated with the expression.

### Subgrammars

Complex grammars can contain references to *subgrammars*. Subgrammars let you modularize your grammar descriptions. For example, this version of the `PrimaryColors` grammar rule refers to the subgrammar `Shades`.

```ruby
PrimaryColors [
  (?Shades [red pink burgundy])
    { <color red> }
  (?Shades [(?sky blue) turquoise])
    { <color blue> }
  (?Shades [(?forest green chartreuse])
    { <color green> }

Shades [
  dark
  light
]
```

One grammar references another by its rule name; the result is the same as if the referenced rule’s grammar expression appeared in place of the rule name. Thus, the preceding `PrimaryColors` grammar rule is equivalent to:

```ruby
PrimaryColors [
  (?[dark light] [red pink burgundy])
    { <color red> }
  (?[dark light] [(?sky blue) turquoise])
    { <color blue> }
  (?[dark light] [(?forest green chartreuse])
    { <color green> }
]
```

A subgrammar can itself reference another subgrammar. Thus, you can create hierarchies of grammars.

A reference to a subgrammar can specify a variable to be set to the value returned from the subgrammar; the subgrammar can return a value with an assignment commands of the form:

```ruby
{return (value)}
```
You specify a variable by following the subgrammar name with a colon (:) and a
variable name. You get use the value of the variable by preceding the variable name
with a dollar sign ($). For example:

```plaintext
Flight
  ( [from leaving] City:frCity ) { <origin $frCity> }
  ( [to (arriving in)] City:toCity ) { <destination $toCity> }
]

City
  atlanta { <return "Atlanta"> }
  chicago { <return "Chicago"> }
  dallas { <return "Dallas"> }
```

Field Grammars

For grammars contained in `<field>` elements, only one field item variable can be
set by the grammar. As a consequence, you can choose the slot name arbitrarily.

The following grammar rule identifies month names and sets the month field to the
corresponding three-letter month abbreviation.

```plaintext
Month
  january {<month jan>}
  february {<month feb>}
  march {<month mar>}
...
```

Suppose this rule is in a field grammar and the user says “February”. Then the field
item variable is filled with "feb", the value associated with the grammar expression
for the utterance “February”.

If the field is named `date` (that is, the field has the attribute `name="date"`), you can
examine the field item variable in the field’s `<filled>` element with:

```plaintext
<value expr="date">
```

If the field also has attribute `slot="foo"`, the date variable still evaluates to "feb".
A `<field>` element’s slot attribute and name attribute are both ignored for field
level grammars.

If the grammar description is a disjunction of a set of simple words, you can use a
shorthand notation for field grammars that omits explicit assignment commands, as
shown in the following example. Any matching word will be assigned to a default slot
name.

```plaintext
[ january february march april may june july
  august september october november december
]
```

If a field grammar contains this grammar expression and the user says “February”,
then the field item variable will be assigned the value “february”.

Form Grammars

If you are writing a grammar for a mixed-initiative form, you may use multiple slot
names, each identifying a field item variable to be filled. In this case you can name
each field according to the corresponding slot name in the grammar.
Suppose the following rule is defined as the grammar for a mixed-initiative form. If the user says, “May I talk to a person?” and the form contains a field named support, then <value expr="support"> will evaluate to "true".

Help [  
    (i don’t understand)  
    {<help "true">}  
    [what huh help]  
    {<help "true">}  
    (?{talk [to with]}[someone(a ?live person)])  
    {<support "true">}  
    {{good bye) asta_la_vista}  
    {<exit "true">}  
]

If the field names aren’t the same as the grammar’s slot names, you can use the slot attribute on the <field> elements to map the grammar slot values to the appropriate field item variables. To show this, suppose the preceding grammar is contained in a mixed initiative form that contains a field named x with the attribute slot="support". If the users says, “May I talk to a person?” then <value expr="x"> will evaluate to "true".

Note that only the field’s name attribute represents a variable; the slot attribute is simply a mapping from the grammar assignment command to a particular field. As a result, in this case, <value expr="support"> evaluates to "undefined".

Ambiguous Grammars

A grammar is ambiguous if more than one rule that can match a given user utterance. Ambiguous grammars can be a problem if the different rules make different slot assignments. For example:

Cities [  
    (portland ?maine) {<city Portland> <state Maine>}  
    (portland ?oregon) {<city Portland> <state Oregon>}  
    (dallas ?texas) {<city Dallas> <state Texas>}  
]

The Cities rule is ambiguous because the utterance “Portland” can match two rules; the state slot could be filled either with “Maine” or “Oregon”.

In general, you should avoid using ambiguous grammars. If you choose to use them, you need to enable recognition of multiple interpretations of the user’s speech and implement a mechanism to get user clarification for ambiguous utterances. See Multiple Recognition Results in VoiceXML (http://cafe.bevocal.com/docs/NBest/index.html).

Independent Grammars and Rulesets

A GSL grammar can contain either of the following:

- An independent grammar to be used as an active grammar. An independent grammar consists of a root grammar rule, and possibly additional rules that are used as subgrammars in a grammar hierarchy whose root is the root grammar rule.

- A rule set containing a number of named rules that can be used as subgrammars by other grammar rules.

The ruleset attribute of the <grammar> element determines whether the grammar specifies an independent grammar or a rule set. If ruleset is "false", ...
which is the default, the <grammar> element specifies an independent grammar; if ruleset is "true", the element specifies a ruleset.

Referencing Grammars

A VoiceXML application references a grammar with the <grammar> element. The grammar can be either:

- An inline grammar description (included directly in your VoiceXML code).
- An external grammar described in a separate file.

Using an Inline Grammar

To include a grammar description directly in your VoiceXML code, embed it inside the grammar tag. You can use CDATA escapes when you want to include special characters not normally permitted in XML documents, such as < and >.

This example shows an inline form grammar:

```xml
<form>
  <grammar>
    <![CDATA[
      (atlanta ?georgia) { <city Atlanta> <state Georgia> }
      (chicago ?illinois) { <city Chicago> <state Illinois> }
      (dallas ?texas) { <city Dallas> <state Texas> }
    ]]>}
  </grammar>
  ...
</form>
```

If the grammar contains more than one rule and the ruleset attribute is "false", the first rule is assumed to be the root rule of the independent grammar.

Referencing an External Grammar

To reference a grammar description that is contained in an external file, set the src attribute of the <grammar> element to the URL of the grammar file.

You can specify the URL of the grammar file in either absolute or relative terms. For example, a VoiceXML document http://myCompany.com/myvxml.vxml could use any of the following URLs to refer to the same grammar file:

- Absolute
  "http://myCompany.com/vxml/mygram.grammar"
- Relative to the host:
  "/vxml/mygram.grammar"
  Note the initial forward slash.
- Relative to the location of the VoiceXML document:
  "vxml/mygram.grammar"
  Note the lack of the initial forward slash.
In addition, you can use an at sign (@) in the URL to refer to the BeVocal hosting platform. For example:

"@/bevocal/grammars/numbers.grammar"

**Independent Grammar**

If the ruleset attribute is "false", the `<grammar>` element must specify an independent grammar; in that case, the `src` attribute consists of the URL for the external grammar file followed a pound sign (#) and the name of the root rule of the grammar. For example, suppose the file `colors.grammar` contains the following three rules.

ShadeAndColor (Shades Colors)

Colors [  
  red {<color red>}  
  blue {<color blue>}  
]

Shades [  
  dark {<shade dark>}  
  light {<shade light>}  
]

(Note that the `<grammar>` tags and CDATA escapes must not be included in the grammar description given in the external file.)

The following field could use the `Colors` rule as the root (and only) rule of its grammar:

```xml
<field name="color">
  <grammar src="colors.grammar#Colors" />
...
</field>
```

**Rule Set**

If the ruleset attribute is "false", the `<grammar>` element must specify rule set; in that case, the `src` attribute consists of just the URL for the external grammar file. For example:

```xml
<form>
  <grammar ruleset="true" src="colors.grammar" />
...
</form>
```
Grammar Scopes

VoiceXML grammars are scoped. By default, the scope of a grammar is set by the element that contains the grammar. There are four basic scopes:

<table>
<thead>
<tr>
<th>Scope</th>
<th>Grammar is in scope when execution is:</th>
<th>Default scope for a grammar defined in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>In the <code>&lt;field&gt;</code> where it is defined or referenced.</td>
<td>A <code>&lt;field&gt;</code> element.</td>
</tr>
<tr>
<td>Dialog</td>
<td>In the dialog where it is defined or referenced.</td>
<td>A <code>&lt;form&gt;</code> element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A <code>&lt;choice&gt;</code> element.</td>
</tr>
<tr>
<td>Document</td>
<td>In the document where it is defined or referenced.</td>
<td>A <code>&lt;link&gt;</code> element directly under a <code>&lt;vxml&gt;</code> element.</td>
</tr>
<tr>
<td>Application</td>
<td>Anywhere in the application.</td>
<td>Any element with document scope that appears in the application root document.</td>
</tr>
</tbody>
</table>

The scope of an independent grammar determines when the grammar is active; the scope of a rule set determines whether the rules it contains are used by referencing rules.

Independent Grammars

An independent grammar is active when it is in scope. The speech-recognition engine will recognize utterances from any and all active grammars. If the user says something that matches a grammar of higher scope, control jumps to the higher-level element that contains the matching grammar. In the case of an event handler, control resumes in the original dialog after the event is handled.

In general, we recommend that you keep the scope of your grammars as narrow as possible, which allows the speech-recognition engine to be most efficient.

Rule Sets

The rules in a rule set are used only when the rule set is in scope. If a grammar rule A refers to a rule B from a rule set that is not in scope, the effect is the same as if A did not refer to B.

For example, the Color rule is in a ruleset:

```
<grammar ruleset="true">
  Color [red blue yellow]
  Size [small medium large]
</grammar>
```

The rule Item refers to the rule Color:

```
<grammar>
  Item [
    statue
    ( Color [ book ball car ] )
  ]
</grammar>
```
When the rule set containing Color is in scope, the Item rule is equivalent to:

```
Item[
  statue
  ( [red blue yellow] [ book ball car ] )
]
```

However, when the rule set containing Color is not in scope, the Item rule is equivalent to:

```
Item[
  painting
  ( [ book ball car ] )
]
```

### Grammar Syntax

Grammar descriptions consist of rule names (optional), grammar expressions (required), and assignment commands (optional). For example:

```
Colors[
  (?[light dark] red) { <color red> }
  (?[light dark] blue) { <color blue> }
  (?[light dark] green) { <color green> }
]
```

### Rule Names

Every rule name must start with an upper-case letter. You can use the following characters within rule names:

- upper case or lower case letters (but the rule name cannot be all uppercase letters)
- digits
- special characters, limited to:
  - (hyphen), _ (underscore), ‘ (single quote), @ (at sign), . (period)

### Grammar Expressions

Grammar expressions consist of word tokens, rule references, and grammar operators. Word tokens correspond to the actual words a user might speak. Rule references correspond to subgrammars.

Word tokens and rule references must be separated from each other by whitespace. Whitespace is optional between the grammar operators and their operands (word tokens or rule names). You can add comments using the semicolon (;) character; everything on the line after a semicolon is ignored.

**Word Tokens**

You can use the following characters within word tokens, without requiring double quotes:

- lower case letters only
- digits
• special characters, limited to:
  - (hyphen), _ (underscore), ’ (single quote), @ (at sign), . (period)

If you enclose a word token in double quotes, you can use other special characters as well, with the exception of whitespace characters. For example, "new^york" is a valid word token, but "new york" is not.

Rule References

A rule reference has one of the following two forms:

• ruleName
• ruleName:variableName

Both forms reference the rule named ruleName as a subgrammar. The second form additionally specifies a variable named variableName to be set to any value returned from the subgrammar.

A grammar expression can reference the following rules:

• Any grammar rule defined in the grammar that contains the grammar expression.
• Any grammar rule in any rule set in that is in scope when the grammar expression is used.

Grammar Operators

Word tokens are rule references are combined into grammar expressions by grammar operators. In the table below, a, b, c, and d represent word tokens, rule references, or more complex grammar expressions:

<table>
<thead>
<tr>
<th>Grammar Operator</th>
<th>Example Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>() sequence</td>
<td>(a b c ... d)</td>
<td>a and b and c and ... and d (in that order)</td>
</tr>
<tr>
<td>[] disjunction</td>
<td>[a b c ... d]</td>
<td>One of a or b or c or ... or d</td>
</tr>
<tr>
<td>? optional</td>
<td>?a</td>
<td>a is optional</td>
</tr>
<tr>
<td>+ positive closure</td>
<td>+a</td>
<td>One or more repetitions of a</td>
</tr>
<tr>
<td>* kleene closure</td>
<td>*a</td>
<td>Zero or more repetitions of a</td>
</tr>
</tbody>
</table>

The following table list some example grammar expressions and gives the set of possible spoken phrases that each expression represents.

<table>
<thead>
<tr>
<th>Grammar expression</th>
<th>Matching utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>[large regular]</td>
<td>• “large”</td>
</tr>
<tr>
<td></td>
<td>• “regular”</td>
</tr>
<tr>
<td>(large regular) fries</td>
<td>• “large fries”</td>
</tr>
<tr>
<td></td>
<td>• “regular fries”</td>
</tr>
<tr>
<td>([large regular] fries)</td>
<td>• “large French fries”</td>
</tr>
<tr>
<td></td>
<td>• “regular French fries”</td>
</tr>
<tr>
<td>([large regular] (?french fries))</td>
<td>• “large fries”</td>
</tr>
<tr>
<td></td>
<td>• “large fries”</td>
</tr>
<tr>
<td></td>
<td>• “regular fries”</td>
</tr>
</tbody>
</table>
Transition Weights

You can assign transition weights, or probabilities, to most grammar expressions using the \textasciitilde\ operator. You specify a probability for a grammar expression as follows:

\texttt{grammarExpression}\textasciitilde\texttt{probability}

The probability must be a non-negative number. The following expression means that a has a 40\% probability of occurring:

\texttt{a\textasciitilde.4}

Probabilities are mostly used in disjunct (OR) constructs. For example:

\texttt{Softdrinks [}
\texttt{  \hspace{0.3cm} coke\textasciitilde.5}
\texttt{  \hspace{0.3cm} sprite\textasciitilde.3}
\texttt{  \hspace{0.3cm} orange\textasciitilde.2}
\texttt{]}

All probabilities in the disjunct are normalized to add up to 1.0. If you don’t specify any probabilities, all expressions in the disjunct are considered to be equally likely.

DTMF Tokens

You can express valid touchtone sequences in a grammar expression using the following DTMF notation:

<table>
<thead>
<tr>
<th>Key Press</th>
<th>DTMF Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dtmf-0</td>
</tr>
<tr>
<td>1</td>
<td>dtmf-1</td>
</tr>
<tr>
<td>2</td>
<td>dtmf-2</td>
</tr>
<tr>
<td>3</td>
<td>dtmf-3</td>
</tr>
<tr>
<td>4</td>
<td>dtmf-4</td>
</tr>
<tr>
<td>5</td>
<td>dtmf-5</td>
</tr>
<tr>
<td>6</td>
<td>dtmf-6</td>
</tr>
<tr>
<td>7</td>
<td>dtmf-7</td>
</tr>
<tr>
<td>8</td>
<td>dtmf-8</td>
</tr>
<tr>
<td>9</td>
<td>dtmf-9</td>
</tr>
<tr>
<td>*</td>
<td>dtmf-star</td>
</tr>
<tr>
<td>#</td>
<td>dtmf-pound</td>
</tr>
</tbody>
</table>
The following grammar expression allows either spoken or telephone keypad input:

\[((john\s+smith)(dtmf-4\sdtmf-9\sdtmf-7\sdtmf-3))\]
\{"<emp\s+john\_smith>\}

The user can either say “John Smith” or use the touchtone sequence 4973 to assign a value of "john_smith" to a field item variable.

**Grammar Assignment Commands**

You use grammar *assignment commands* to attach an assignment operation to a particular grammar expression. The assignment commands for a grammar expression are enclosed in curly braces and immediately follow the grammar expression.

The curly braces can enclose one or more individual assignment commands. An individual assignment command is either a slot-filling command or a return command:

- **A slot-filling command** specifies a value for a field item variable; it has the form:
  \(<\text{slotname}\s+\text{value}>\)
  The slot name indicates the field item variable to be assigned a value.

- **A return command** returns a value from a subgrammar; it has the form:
  \{"\text{return}\(\text{(value)}\)\}
  A return command is useful when the rule reference to the subgrammar specifies a variable. When the subgrammar is matched, the corresponding variable is set to the value returned by the return command.

  **Note:** A grammar rule can include a return command only if it is always used as a subgrammar. If the rule is ever used as a root grammar rule it must not include a return command.

**Value Expressions**

The *value* expression in an assignment command can be a literal value, a variable expression, or a function call.

**Literal Values**

A *literal value* is an integer or a quoted or unquoted string. A literal value is evaluated as follows:

- A quoted string evaluates to itself.
- An integer evaluates to its integer value.
- An unquoted strings that has no integer interpretation evaluates to itself.

As the preceding list suggests, values that can be interpreted as integers are treated as integers. For example, the value 01 in the following grammar rule can be interpreted as an integer:

```
DigitValue [ 
   ([zero\s+oh]\s+one) {<return(01)>} 
   ]
```...

If the user says "oh one", the DigitValue rule will return the integer 1.
If you want an integer value to be treated as a string, you can enclose it in double, for example:

```
DigitString [ ...
  ([zero oh] one) {<return("01")>}
  ...
]
```

If the user says "oh one", the DigitString rule will return the string "01".

**Variable Expression**

If a grammar includes a rule references to subgrammar and that rule reference specifies a variable `variableName`, an assignment command of the containing grammar can use a variable expression of the form:

```
$variableName
```

This expression evaluates to the value returned from the referenced rule.

A special variable, `string`, is used to capture the portion of the utterance that matched a grammar expression; its value can be obtained with expression:

```
$string
```

For example, the following grammar accept the name of a day of the week and sets the `day` field item variable to the word that the user said:

```
DayOfWeek [
  sunday monday tuesday wednesday
  thursday friday saturday
] { <day $string>}
```

**Function Calls**

You can use a *function call* to an arithmetic or string function to compute the value in an assignment command. The parameters to the functions can be value literals, variable expressions, or other function calls.

The following integer and string functions are available for use in value expressions. The third column indicates the default value for a parameters that is undefined.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Default Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>Adds two integers.</td>
<td>0</td>
</tr>
<tr>
<td>sub</td>
<td>Subtracts the second integer from the first.</td>
<td>0</td>
</tr>
<tr>
<td>mul</td>
<td>Multiplies two integers together.</td>
<td>1</td>
</tr>
<tr>
<td>div</td>
<td>Divides the first integer by the second and returns the truncated result.</td>
<td>0 for first parameter; 1 for second parameter</td>
</tr>
<tr>
<td>neg</td>
<td>Returns the negative or positive inverse of an integer.</td>
<td>0</td>
</tr>
<tr>
<td>strcat</td>
<td>Concatenates two strings, but may be nested to effectively concatenate any number of strings, for example: <code>strcat($a1 strcat($a2 $a3))</code></td>
<td>&quot; &quot; (empty string)</td>
</tr>
</tbody>
</table>
The following example illustrates how the functions can be used to compute the value of the `number` field item variable from values returned by subgrammars.

TwoDigit [  
  (Digit:n1 Digit:n2)    
    { <number add((mul(10 $n1)) $n2) }  
  TeenAndTen:n        
    { <number $n> }  
  (DecadeFromTwenty:n1 ?NonZeroDigit:n2)  
    { <number add($n1 $n2)> }  
]

Digit [  
  [zero oh]       { return(0) }  
  NonZeroDigit:d { return($d) }  
]

NonZeroDigit [  
  one   { return(1) }  
  two   { return(2) }  
  …  
  nine  { return(9) }  
]

TeenAndTen [  
  ten   { return(10) }  
  eleven { return(11) }  
  …  
  nineteen { return(19) }  
]

DecadeFromTwenty [  
  twenty { return(20) }  
  thirty { return(30) }  
  …  
  ninety { return(90) }  
]

Notice in the TwoDigit rule, the third grammar expression has an optional element. If the user said “twenty”, the value of \( n_2 \) would be undefined. Since the default value for the \( \text{add} \) function is zero, the result will be correct \( (20 + 0) \).
This chapter describes the XML Speech Recognition Grammar Format.

XML Grammar Basics

An XML grammar, like XML and VoiceXML, uses markup tags and plain text. A tag is a keyword enclosed by the angle bracket (< and >) characters. A tag may have attributes inside the angle brackets. Each attribute consists of a name and a value, separated by an equal (=) sign; and the value must be enclosed in quotes.

Tags occur in pairs; corresponding to the start tag <keyword> is the end tag </keyword>. Between the start and end tag, other tags and text may appear. Everything from the start tag to the end tag, is called an element. If one element contains another, the containing element is called the parent element of the contained element. The container element is called a child element of its containing element. The parent element may also be called a container.

If an element contains no child elements, You can omit the end tag by replacing the final ">"> of the start tag with " />".

Rules

An XML grammar consists of rules. Each rule specifies some user input that can be recognized and, optionally, specifies how to interpret a valid expression in terms of values for field item variables.

The <rule> tag defines a rule in an XML grammar. Each rule definition has a name, specified by the id attribute. A rule’s name must be unique within the scope of the grammar that contains the rule. A rule name names must be a legal XML ID.

A rule’s scope attribute indicates where it can be used. The scope may be either:

- "private" - local to the grammar that contains it.
- "public" - available to be referenced from other grammars.

A rule with no scope attribute is private by default.

Rules specify user input with tokens are rule references.

- A token corresponds to a word that a user might actually speak. Any unmarked text is a token. A token that contains whitespace or other special characters can be enclosed in double quotes. Alternatively, the token can be contain in a <token> element.
- A rule reference refer to another grammar rule specifying user input. The <ruleref> tag references another rule.
Inline and External Grammars

An inline XML grammar consists of one or more `<rule>` elements inside the VoiceXML `<grammar>` tag.

An external XML grammar is an XML file with a `<grammar>` element containing `<rule>` elements. Like any XML file, an external XML grammar file begins with the header:

```xml
<?xml version="1.0" ?>
```

Note that the version number refers to the version of XML, not the version of the grammar it contains.

Default Rule

Every XML grammar has a default rule. If the `<grammar>` element containing the rule has a `root` attribute, that attribute names the default rule. Otherwise, the default rule is constructed implicitly as the logical OR (disjunction) of all public rules in the grammar.

Language and Pronunciation

The `xml:lang` attribute of a `<grammar>` element specifies its language of the spoken input. This alerts the speech-recognition engine should use pronunciation rules, phonetic inventory and acoustic models for the specified language.

Various components within a grammar can have a `lang-list` attribute, which specifies one or more languages in which the spoken input is expected. A single language is analogous to the language for the grammar. If multiple languages are specified, the speech-recognition engine should use the pronunciation, phonetic inventory and acoustic models of the different languages in parallel.

The language is specified with an identifier that designates the language and, optionally, the country whose local pronunciation and vocabulary should be used. For example, the identifier `en-US` designated United States English.

Currently, English is the only supported language.

Tag Summary

The following tags can be used in XML grammars:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;grammar&gt;</code></td>
<td>Defines the grammar in an external XML grammar file.</td>
</tr>
<tr>
<td><code>&lt;rule&gt;</code></td>
<td>Defines a grammar rule.</td>
</tr>
<tr>
<td><code>&lt;token&gt;</code></td>
<td>Input element that specifies words to be spoken by the user.</td>
</tr>
<tr>
<td><code>&lt;ruleref&gt;</code></td>
<td>Input element that references another rule.</td>
</tr>
<tr>
<td><code>&lt;one-of&gt;</code></td>
<td>Input element that indicates alternative user inputs.</td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td>Input element that indicates optional or repeated user input.</td>
</tr>
<tr>
<td><code>&lt;tag&gt;</code></td>
<td>Specifies how to interpret the user input.</td>
</tr>
<tr>
<td><code>&lt;example&gt;</code></td>
<td>Example phrase that matches the containing grammar rule.</td>
</tr>
</tbody>
</table>

The remaining sections of this chapter describe these tags in alphabetical order.
Example phrase that matches the containing grammar rule.

Syntax
<example>
Example Input
</example>

Description
An <example> element encloses a sequence of tokens corresponding to user input that matches the containing rule. It is illustrative, for the benefit of a developer reading the grammar; the speech-recognition engine ignores the element.

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule&gt;</td>
<td>None.</td>
</tr>
</tbody>
</table>
<grammar>

Defines the grammar in an external XML grammar file.

Syntax

```
<grammar
    version="1.0"
    xml:lang="language"
    mode="voice"|"dtmf"
    root="string" >
  Rules
</grammar>
```

Description

Top-level element in each XML grammar file.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>XML Speech Recognition Grammar Format version used in this document. <em>Optional</em> (default is &quot;1.0&quot;). The only accepted value is &quot;1.0&quot;.</td>
</tr>
<tr>
<td>xml:lang</td>
<td>The language identifier for the grammar; see “Language and Pronunciation” on page 22. <em>Optional</em> (default is &quot;en-US&quot;) The accepted language identifiers are: • en - English • en-US - United States English</td>
</tr>
<tr>
<td>mode</td>
<td>The mode of the contained or referenced grammar. <em>Optional</em> (default is &quot;voice&quot;). • voice - Spoken input • dtmf - <em>Not implemented</em>. DTMF input.</td>
</tr>
<tr>
<td>root</td>
<td>The name of the explicit default grammar rule. <em>Optional</em> (if omitted, an implicit default rule is used; see “Default Rule” on page 22).</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>&lt;rule&gt;</td>
</tr>
</tbody>
</table>
Input element that indicates optional or repeated user input.

**Syntax**

```xml
<item
    repeat="?"|"*"|"+"|"optional"|"M-N"|"M-" >
  Content
</item>
```

**Description**

An `<item>` element can contain any number of input elements and tag elements. The input elements indicate a sequence that must be matched in order. The `repeat` attribute applies to the entire sequence, indicating that it is optional or that it may be repeated.

Any contained `<tag>` elements apply to the entire optional or repeated sequence. If the user input matches the input elements and `repeat` attribute, the `<tag>` elements are interpreted to assign values to field item variables.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat</td>
<td>Indicates the number of times that the contained expansion may be repeated. Optional (if omitted, the sequence of input elements must occur exactly once).</td>
</tr>
<tr>
<td></td>
<td>• <code>?</code> - The item is optional.</td>
</tr>
<tr>
<td></td>
<td>• <code>*</code> - The item is can occur zero or more times.</td>
</tr>
<tr>
<td></td>
<td>• <code>+</code> - The item can occur one or more times.</td>
</tr>
<tr>
<td></td>
<td>• <code>optional</code> - The item is optional.</td>
</tr>
<tr>
<td></td>
<td>• <code>M-N</code> - The item can occur between <code>M</code> and <code>N</code> times (inclusive). <code>M</code> and <code>N</code> must both be nonnegative integers and <code>M</code> must be less than or equal to <code>N</code>.</td>
</tr>
<tr>
<td></td>
<td>• <code>M-</code> - The item can occur <code>M</code> or more times. <code>M</code> must be a nonnegative integer.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;rule&gt;</code></td>
<td><code>&lt;token&gt;</code></td>
</tr>
<tr>
<td><code>&lt;rule&gt;</code></td>
<td><code>&lt;ruleref&gt;</code></td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td><code>&lt;item&gt;</code></td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td><code>&lt;one-of&gt;</code></td>
</tr>
<tr>
<td><code>&lt;item&gt;</code></td>
<td><code>&lt;tag&gt;</code></td>
</tr>
</tbody>
</table>
<one-of>

Input element that indicates alternative user inputs.

Syntax

\[
\text{<one-of} \\
\hspace{1em} \text{lang-list="languages"} > \\
\hspace{1em} \text{Alternatives} \\
\hspace{1em} \text{</one-of>}
\]

Description

The contained items are alternatives; any one of them may be matched by the user input. Each alternative is an \text{<item>} element.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lang-list</td>
<td>A comma-separated list of the identifiers for languages in which the alternative inputs may be spoken; see &quot;Language and Pronunciation&quot; on page 22. Optional (default is &quot;en-US&quot;)</td>
</tr>
<tr>
<td></td>
<td>The accepted language identifiers are:</td>
</tr>
<tr>
<td></td>
<td>- en - English</td>
</tr>
<tr>
<td></td>
<td>- en-US - United States English</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule&gt;</td>
<td>&lt;item&gt;</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td></td>
</tr>
</tbody>
</table>
<rule>

Defines a grammar rule.

Syntax

```xml
<rule
    id="string"
    scope="private" | "public" >
    Other Content
</rule>
```

Description

An `<rule>` element can contain any number of input elements, examples, and tag elements. The input elements indicate a sequence that must be matched in order.

Any contained `<tag>` elements apply to the entire sequence. If the user input matches the input elements, the `<tag>` elements are interpreted to assign values to field item variables.

Any `<example>` elements are ignored by the speech-recognition engine.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The name of the rule; must be unique within the containing grammar (this is enforced by XML). The rule must not be the name of one of the special rules &quot;NULL&quot;, &quot;VOID&quot;, or &quot;GARBAGE&quot;.</td>
</tr>
</tbody>
</table>
| scope | The scope in which this rule can be used. Optional (default is "private").

- private - The rule can be used only by the containing grammar
- public - The rule can be reference by another grammar (in a `<ruleref>` element in the referencing grammar).

**Note:** Do not confuse the scope of a rule with the scope of a containing grammar. The scope of the grammar indicates where in the VoiceXML application the grammar is active.

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;grammar&gt;</code></td>
<td><code>&lt;token&gt;</code>&lt;br&gt;<code>&lt;ruleref&gt;</code>&lt;br&gt;<code>&lt;item&gt;</code>&lt;br&gt;<code>&lt;one-of&gt;</code>&lt;br&gt;<code>&lt;tag&gt;</code>&lt;br&gt;<code>&lt;example&gt;</code></td>
</tr>
</tbody>
</table>
**<ruleref>**

Input element that references another rule.

**Syntax**

```
<ruleref
    lang-list="languages"
    special="NULL" | "VOID" | "GARBAGE"
    uri="URI" >
Optional Tags
</ruleref>
```

**Description**

The referenced rule specifies user input to be matched.

If the user input matches the referenced rule, any contained `<tag>` elements are interpreted to assign values to field item variables.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lang-list</td>
<td>The language and optional country local identifier for the referenced rule. <em>Optional</em> (default is &quot;en-US&quot;)</td>
</tr>
<tr>
<td>special</td>
<td>The referenced special rule. <em>Optional</em> (as an alternative to uri).</td>
</tr>
<tr>
<td></td>
<td>- NULL - Rule that is automatically matched, that is, matched without the user speaking any word.</td>
</tr>
<tr>
<td></td>
<td>- VOID - Rule that can never be spoken. Inserting VOID into a sequence automatically makes that sequence unspeakable</td>
</tr>
<tr>
<td></td>
<td>- GARBAGE - Rule rule that matches any speech up until the next rule match, the next token or until the end of spoken input.</td>
</tr>
<tr>
<td>uri</td>
<td>The URI of the referenced rule. <em>Optional</em> (as an alternative to special). May be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- #ruleName - References the local rule ruleName in the contained grammar.</td>
</tr>
<tr>
<td></td>
<td>- grammarFileURI#ruleName - References the public rule ruleName in the grammar defined in the XML grammar file whose URI is grammarFileURI.</td>
</tr>
<tr>
<td></td>
<td>- grammarFileURI - References the default rule of the grammar defined in the XML grammar file whose URI is grammarFileURI. See “Default Rule” on page 22.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule&gt;</td>
<td>&lt;tag&gt;</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td></td>
</tr>
</tbody>
</table>
<tag>

Specifies how to interpret the user input.

Syntax

<tag>
  slotName="value"
</tag>

Description

Within a <tag> element, slotName identifies the field item variable to be assigned the specified value when a user utterance matches the input specified by the containing element.

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule&gt;</td>
<td>None.</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td></td>
</tr>
</tbody>
</table>
<token>

Input element that specifies words to be spoken by the user.

Syntax

<token
    lang-list="languages">
    Content
</token>

Description

If the user input matches the contained words, any contained <tag> elements are interpreted to assign values to field item variables.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lang-list</td>
<td>A comma-separated list of the identifiers for languages in which the user input may be spoken; see &quot;Language and Pronunciation&quot; on page 22. <em>Optional</em> (default is &quot;en-US&quot;)</td>
</tr>
<tr>
<td></td>
<td>The accepted language identifiers are:</td>
</tr>
<tr>
<td></td>
<td>• en - English</td>
</tr>
<tr>
<td></td>
<td>• en-US - United States English</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rule&gt;</td>
<td>&lt;tag&gt;</td>
</tr>
<tr>
<td>&lt;item&gt;</td>
<td></td>
</tr>
</tbody>
</table>