

# **PHILSOFT**

## **Detailed specification file**

This document briefly describes the detailed specification file used to specify a recognition model for PHILSOFT ([a detailed version is available in French](#)). This document focuses on the keywords that are available to specify the model, and also on the standard order of the blocs of data. This information is provided through an [outline of the detailed specification](#) file. The [outline of the specification library file](#) is also given in this document as this file shares many points with the detailed specification file. In fact the library file is used to specify everything which is general (for example, phonemes, rejection units, ...) whereas the detailed specification file is used to specify everything which is related to the current model only.



## Table of contents

<b>1. ABOUT THE SPECIFICATION FILE.....</b>	<b>5</b>
<b>2. SPECIAL LINES AVAILABLE .....</b>	<b>7</b>
2.1 STANDARD COMMAND LINES.....	7
2.2 MISCELLANEOUS COMMANDS .....	7
2.3 LISTING OF SPECIFIC INFORMATION .....	7
2.4 COMMAND BLOC .....	9
<b>3 OUTLINE OF THE DETAILED SPECIFICATION FILE (.DEF).....</b>	<b>11</b>
3.1 SYNTACTIC SPECIFICATIONS.....	11
3.2 LEXICAL SPECIFICATIONS.....	13
3.3 ACOUSTIC SPECIFICATIONS.....	14
<b>4 OUTLINE OF THE SPECIFICATION LIBRARY FILE (.XDF).....</b>	<b>15</b>
4.1 INITIALIZATION DATA.....	15
4.2 MODIFICATION DATA.....	17



## 1. About the specification file

The specification of a speech recognition model is done using a text file, which can be editing with any text editor.

A few rules must be followed. Currently each line should not be longer than 255 characters. However this is not a harmful limitation as lines may be split almost anywhere (as in the C programming language).

Some special lines, starting with '#' as the first non-space character, are available. They are described in the following section, and are mainly used to define symbols, include characters or list required information. This last feature was introduced and used when developing the program.

The specifications appear as blocs of data. Each bloc is identified through one or several keywords followed by an opening brace '{' and ends by a closing brace '}'. The corresponding data is placed between the braces, and may contain other blocs of data.

Some pieces of information are specified as lists. They are also identified by one or several keywords, followed by the equal sign '=', followed by the corresponding data and ends by a semi colon ';'. If the data is made of several elements, parenthesis '(' and ')' should be used, and the various elements must be separated by comas ','.

As an efficient modeling of the phonemes requires a proper handling of the left and right contexts, a context specification is available in some blocs. This is achieved using the square brackets '[' and ']' or the braces '(' and ')'. The first form allows manipulations of sets of units (usually phonemes) previously defined. The second form handles only lists of units.

The tokens of the specification are usually character strings. They may be enclosed between double quotes. The double quotes are mandatory if the string contains non-alphanumeric characters other than "\$" (dollar) and "\_" (underscore). For example, in French, double quotes must be used for string containing accented letters such as 'é', 'ê'... They must also be used if the string contains space characters.

All the space characters that do not belong to strings are meaningless. They are simply used to separate the tokens.

The exclamation mark '!' is used to introduce a comment. The remaining part of the line (i.e. all the characters after and including the exclamation mark) is ignored.

Lines starting with '#' as the first non-space character are treated as special lines.



## 2. Special lines available

The special lines undergo a specific treatment. They are mainly used to define synonyms, to include the data of other files or do some other stuff.

### 2.1 Standard command lines

**#include** "File\_Name"

Include at this place all the data from the specified file. When all the data from the included file is treated, the next line of the current file will be read. Note that the included file may also contain such an instruction. The double quotes around the file name are mandatory.

**#define** Token Replacement\_String

This allows the definition of synonyms.

**#undef** Token

This command removes the defined token from the list.

### 2.2 Miscellaneous commands

**#Exit**

This command stops the compilation of the model. Was used when developing the software.

**#Listing** [Keyword [Keyword [Pattern]]]

This command allows listing some pieces of information. It was also useful when developing the program. The required information is specified through keywords. The pattern may be used to specify which elements will be listed.

**#Purge** [Data\_Type]

This command checks the data of a given type (syntactic, lexical or acoustic) and is operational only in the bloc "Commandes".

**#Compilation** [Level]

This command is also operational only in the bloc "Commandes". It implies the compilation of the model network at the required level: syntactic (i.e. at word level), lexical (i.e. usually at the phoneme level) or acoustic (i.e. final network).

### 2.3 Listing of specific information

The command for listing specific information is the following:

**#Listing** [First\_Keyword [Second\_Keyword [Pattern]]]

The table below displays the possible values for the keywords. The 4 first columns indicate which type of information is concerned by the keywords. The keywords are case insensitive and can be abbreviated. The pattern limits the listing to the objects having a name that matches that pattern.

The information provided depends on the state of the compilation when the listing command is invoked.

S y n t	L e x i t	A c t i v e	I n f o r m a t i o n	Possible commands (Keywords)	Information given (listing)
#	#	#	#	<b>Sommaire Memoire</b>	Memory size used by the networks, the symbols and the sets (contexts).
#	#	#	#	<b>Status Symboles</b>	State of elements: declared, defined or used.
#			#	<b>Entrees Syntaxiques</b>	Names of defined entry states.
#			#	<b>Application</b>	Main network. Content depends on the compilation level when listing is required.
			#	<b>Regroupements</b>	List the states that have been (safely) merged during network optimization.
#				<b>Elements Syntaxiques</b>	List syntactic elements: main network entry states, non-terminal and terminal elements.
#				<b>Reseaux Syntaxiques</b>	List syntactic networks associated to various syntactic elements.
#			#	<b>Mots Syntaxiques</b>	List syntactic words (output at recognition time).
#			#	<b>Unites Syntaxiques</b>	List syntactic units (specify the modeling).
	#		#	<b>Unites Lexicales</b>	List lexical units (usually phonemes).
	#			<b>Contextes Lexicaux</b>	List contexts defined or used at the lexical level (for phonological rules).
	#			<b>Reseaux Lexicaux</b>	List lexical networks (usually phonetic transcriptions of the words).
		#		<b>Noms Modeles</b>	List names of acoustic models that have been defined.
		#		<b>Etats Modeles</b>	List names given to states in acoustic models.
		#		<b>Fonctions Modeles</b>	List names given to pdf in acoustic models.
		#		<b>Contextes Acoustiques</b>	List contexts defined or used at the acoustic level (for defining contextual modeling).
		#		<b>Modeles Acoustiques</b>	List acoustic network of defined models.
		#		<b>Reseaux Acoustiques</b>	List acoustic network of models associated to lexical



					unit (note that a model is first defined, and then applied to one or several lexical units).
			#	Noms Etats	List names given to states in acoustic networks.
			#	Noms Fonctions	List names given to pdf in acoustic networks.

## 2.4 Command bloc

A special command bloc, identified by the keyword "Commandes" is available. Its main purpose was to check and display information while developing the software. It contains only special lines for checking data or compiling the network to a specific level. Listing of information can be obtained (through the "#Listing" command) before or after any of these commands.

```

Commandes {
  #Purge Syntaxe
  #Purge Lexique
  #Purge Acoustique
  #Compilation Syntaxe
  #Compilation Lexique
  #Compilation Acoustique
}

```



### 3 Outline of the detailed specification file (.DEF)

These few pages outline the content of the detailed specification file. They show the standard order of the blocs of data as well as the keywords available. In the current version of the API all the keywords are in French. A few comments are added.

The file contains 3 main blocs. The first one specifies the syntax for the task, the second one describes the words of the vocabulary, and the third one specifies the acoustic modeling.

All the information, which is provided in a library file, is not repeated in this file. As usually the library file specifies standard units (silences, phonemes) and phonological rules, the detailed specification file is often limited to the syntax specification and the description of the vocabulary words.

#### 3.1 Syntactic specifications

```
!
! Specify the syntax of application.
!
Syntaxe {

    ! Declare the names of the terminal elements.
    Elements Terminaux = ( Terminal_Element , ... ) ;

    ! Declare the names of the non terminal elements.
    Elements Non_Terminaux = ( Non_Terminal_Element , ... ) ;

    ! Declare the names of main entry points.
    Entrees Syntaxiques = ( Syntactic_Entry , ... ) ;

    ! Kept for compatibility with older versions.
    ! Should not be used any longer.
    Productions {
        Syntactic_Entry = Syntactic_Expression ;
        Non_Terminal_Element = Syntactic_Expression ;
    }

    ! Standard bloc for declaring and defining the terminal elements
    ! By default Syntactic_Word = Syntactic_Unit = Terminal_Element
    Elements Terminaux {
        Terminal_Element ;                      ! Usual default short form.
        Terminal_Element = < * , * > ;          ! Default form.
        Terminal_Element = < Syntactic_Word , Syntactic_Unit > ;
    }

    ! Standard bloc for declaring and defining some labels (tags) that
    ! define strings to be delivered in addition of recognized words
    ! By default Output_String = Label
    Marqueurs {
        Label ;                                ! Usual default short form.
        Label = < * > ;                          ! Default form.
        Label = < Output_String > ;
    }

    ! Standard bloc for declaring and defining (if necessary)
    ! some labels (tags) that indicate place where to insert
    ! an external model, for example a speaker dependent model.
    ! By default External_Model = Label
```

```

Marqueurs Modeles_Externes {
    Label ;                                ! Usual default short form.
    Label = < * > ;                        ! Default form.
    Label = < External_Model > ;
}

! Standard bloc for declaring and defining non terminal elements.
! They can be defined using syntactic expressions or
! the corresponding network may be described directly.
Elements Non_Terminaux {
    Non_Terminal_Element = Syntactic_Expression ;
    Non_Terminal_Element {
        Nombre Etats = Number_Of_States ;
        Nombre Transitions = Number_Of_Transitions ;
        Transitions {
            < Starting_State, Ending_State, Syntactic_Element > ;
        }
    }
}

! Standard bloc for declaring and defining the main entry states
! of the model network.
! They can be defined using syntactic expressions or
! the corresponding network may be described directly.
Entrees Syntaxiques {
    Entry_State_Name = Syntactic_Expression ;
    Entry_State_Name {
        Nombre Etats = Number_Of_States ;
        Nombre Transitions = Number_Of_Transitions ;
        Transitions {
            < Starting_State, Ending_State, Syntactic_Element > ;
        }
    }
}

! Kept for compatibility with older versions.
! Should not be used any longer.
Substitutions Mots {
    Old_Syntactic_Word = New_Syntactic_Word ;
}

! Kept for compatibility with older versions.
! Should not be used any longer.
Substitutions Unites {
    Old_Syntactic_Unit = New_Syntactic_Unit ;
}

! For the terminal elements, the label and external models,
! * indicates the default value.

! In the syntactic expressions
! + indicates a choice (alternative),
! . indicates a succession of elements,
! * indicates a loop
! ( ... ) a group (a syntactical sub-expression)
! < ...,... > the 2 terms of a terminal element.

```

### 3.2 Lexical specifications

```

!
! Specify the lexical part of the model.
!
Lexique {

    ! List of lexical units used to describe syntactic units,
    ! usually phonemes or word depending on the modeling type.
    Unites = ( Lexical_Units , ... ) ;

    ! Contexts to be used in phonological rules.
    Contextes {
        Lexical_Context = ( Lexical_Unit , ... ) ;
        Lexical_Context = [ Set_Of_Lexical_Units ] ;
    }

    ! Specify the description of each syntactic unit,
    ! usually in term of phonemes.
    Descriptions {
        Syntactic_Unit = Lexical_Expression ;
    }

    ! These phonological rules add pronunciation variants if the
    ! sequence of phonemes occurs in the given context.
    Alternatives {
        [ Left_Context ] . Sequence_Of_Units. [ Right_Context ]
                                = Lexical_Expression_To_Add ;
    }

    ! These phonological rules replace a sequence of phonemes occuring
    ! in a given context by an expression.
    Substitutions {
        [ Left_Context ] . Sequence_Of_Units. [ Right_Context ]
                                = Lexical_Expression_For_Replacement ;
    }
}

! The sets of lexical units are defined by adding (+ sign) or
! removing (- sign) lexical units or other lexical contexts.

! In the lexical expressions
!   + indicates a choice (alternative),
!   . indicates a succession of elements,
!   ( ... ) a group (a lexical sub-expression)

```

### 3.3 Acoustic specifications

```

!
! Specify the acoustic part of the model
!
Acoustique {

    ! Contexts to be used in contextual modeling.
    Contextes {
        Acoustic_Context = ( Lexical_Unit , ... ) ;
        Acoustic_Context = [ Set_Of_Lexical_Units ] ;
    }

    ! Specification of an acoustic model
    Modele Acoustic_Model_Name {
        Nombre Etats = Number_Of_States ;
        Nombre Transitions = Number_Of_Transitions ;
        Nombre Fonctions = Number_Of_Functions ;
        Etats Entrees = ( State_Name , ... ) ;
        Etats Sorties = ( State_Name , ... ) ;
        Etats Internes = ( State_Name , ... ) ;
        Fonctions = ( Pdf_Name , ... ) ;
        Transitions {
            < Starting_State , Ending_State , Associated_Pdf > ;
        }
    }

    ! Indicate which model should be used for each unit
    Modelisations {
        Acoustic_Model_Name
        avec {
            Starting_State = [ Set_Of_Lexical_Units ] ;
            Ending_State = [Set_Of_Lexical_Units ] ;
        } pour ( Lexical_Unit , ... ) ;
    }
}

```

## 4 Outline of the specification library file (.XDF)

These few pages outline the content of the specification library file. They are intended to remind the standard order of the blocs as well as the keywords. In the current version of the API all the keywords are in French. A few comments are added.

Library files are used to store information which is task independent, such as the list of units (usually phonemes), some lexical contexts and phonological rules, some acoustic models, acoustic contexts and contextual modeling of the units (phonemes).

The library file contains 2 main blocs. The first one refers to data, which is process before processing of the detailed specification file. This bloc mainly specifies data and acoustic models. The second bloc is process after the detailed specification file. It mainly contains the phonological rules and indicates which contextual modeling is to be used for each unit.

### 4.1 Initialization data

```
!
! Informations processed before the specification file.
!
Initialisations {                               ! or Avant {

!
! Syntactic initializations.
!
Syntaxe {

! Declare the names of the terminal elements.
Elements Terminaux = (Terminal_Element , ... ) ;

! Declare the names of the non terminal elements.
Elements Non_Terminaux = (Non_Terminal_Element , ... ) ;

! Standard bloc for declaring and defining the terminal elements
! By default Syntactic_Word = Syntactic_Unit = Terminal_Element
Elements Terminaux {
  Terminal_Element ;                               ! Usual default short form.
  Terminal_Element = < * , * > ; ! Default form.
  Terminal_Element = < Syntactic_Word , Syntactic_Unit > ;
}

! Standard bloc for declaring and defining non terminal elements.
! They can be defined using syntactic expressions or
! the corresponding network may be described directly.
Elements Non_Terminaux {
  Non_Terminal_Element = Syntactic_Expression ;
  Non_Terminal_Element {
    Nombre Etats = Number_Of_States ;
    Nombre Transitions = Number_Of_Transitions ;
    Transitions {
      < Starting_State, Ending_State, Syntactic_Element > ;
    }
  }
}
}

}
```

```

!
! Lexical initializations.
!
Lexique {

    ! List of lexical units used to describe syntactic units,
    ! usually phonemes or word depending on the modeling type.
    Units = ( Lexical_Unit , ... ) ;

    ! Contexts to be used in phonological rules.
    Contextes {
        Lexical_Context = ( Lexical_Unit , ... ) ;
        Lexical_Context = [ Set_Of_Lexical_Units ] ;
    }

    ! Specify the description of each syntactic unit,
    ! usually in term of phonemes.
    Descriptions {
        Syntactic_Unit = Lexical_Expression ;
    }
}

```

```

!
! Acoustic initializations.
!
Acoustique {

    ! Contexts to be used in contextual modeling.
    Contextes {
        Acoustic_Context = ( Lexical_Unit , ... ) ;
        Acoustic_Context = [ Set_Of_Lexical_Units ] ;
    }

    ! Specification of an acoustic model
    Modele Acoustic_Model_Name {
        Nombre Etats = Number_Of_States ;
        Nombre Transitions = Number_Of_Transitions ;
        Nombre Fonctions = Number_Of_Functions ;
        Etats Entrees = ( State_Name , ... ) ;
        Etats Sorties = ( State_Name , ... ) ;
        Etats Internes = ( State_Name , ... ) ;
        Fonctions = ( Pdf_Name , ... ) ;
        Transitions {
            < Starting_State , Ending_State , Associated_Pdf > ;
        }
    }
}

```

```

}

```



## 4.2 Modification data

```

!
! Informations processed after the specification file.
!
Modifications {                               ! or Apres {

!
! Syntactic modifications.
!
Syntaxe {

    ! Kept for compatibility with older versions.
    ! Should not be used any longer.
    Substitutions Mots {
        Old_Syntactic_Word = New_Syntactic_Word ;
    }

    ! Kept for compatibility with older versions.
    ! Should not be used any longer.
    Substitutions Unites {
        Old_Syntactic_Unit = New_Syntactic_Unit ;
    }

}

!
! Lexical modifications.
!
Lexique {

    ! Contexts to be used in phonological rules.
    Contextes {
        Lexical_Context = ( Lexical_Unit , ... ) ;
        Lexical_Context = [ Set_Of_Lexical_Units ] ;
    }

    ! These phonological rules add pronunciation variants if the
    ! sequence of phonemes occurs in the given context.
    Alternatives {
        [ Left_Context ] . Sequence_Of_Units. [ Right_Context ]
                                = Lexical_Expression_To_Add ;
    }

    ! These phonological rules replace a sequence of phonemes occuring
    ! in a given context by an expression.
    Substitutions {
        [ Left_Context ] . Sequence_Of_Units. [ Right_Context ]
                                = Lexical_Expression_For_Replacement ;
    }

}

```

```
!  
! Acoustic modifications  
!  
Acoustique {  
  
    ! Contexts to be used in contextual modeling.  
    Contextes {  
        Acoustic_Context = ( Lexical_Unit , ... ) ;  
        Acoustic_Context = [ Set_Of_Lexical_Units ] ;  
    }  
  
    ! Indicate which model should be used for each unit  
    Modelisations {  
        Acoustic_Model_Name  
        avec {  
            Starting_State = [ Set_Of_Lexical_Units ] ;  
            Ending_State = [Set_Of_Lexical_Units ] ;  
        } pour ( Lexical_Unit , ... ) ;  
    }  
}  
  
}
```