synthiam.com

Synbot Plugin Tutorial - interface to Syn Bot Software

How To install bot software, EZ builder Synbot plugin and example SIML project files

Last Updated: 3/20/2018

Introduction : Discover Synthetic Intelligence Network Bot development framework

Synthetic Intelligence Network is a brand of REVARN™ Cybernetics. An Artificial Intelligence Software development company established in Bangalore.
This tutorial introduce you to their Bot development Framework and to the EZ Builder SynBot Plugin I have developped.
It also introduce with example files to a SIML pseudo Framework which can make easier creation of bot dialogs intended for EZ Robots Command , with a multi language approach (English and french in the examples)
It enables you to create and deploy your Bot as a standalone package linked to EZ Builder Software on your own server.
In SIML code examples , characters > and < of SIML tags have been replaced with] and [in order to be displayable in the tutorial.

syn.co.in/
To learn more on SIML :
developer.syn.co.in/tutorial/siml/index.html
To begin :
simlbot.com/

Step 1 : Install Synthetic Intelligence Network software

You need first to download and install Syn Bot Studio software which include the Tools used for creating and updating Bot Dialogs.
Download here simlbot.com/

Install the software
Read the tutorial on Syn Bot studio here : developer.syn.co.in/tutorial/bot-studio/index.html
On this page you can access to all the tutorials including a full description of SIML language used to create Bot dialogs

Step 2 : Create a SIML project and install SIML example files

Follow the instructions of Syn Bot Studio tutorial to create a SIML project.
Start Syn Bot Studio - then File New and Project
Fill in the information on your project - Select Empty for Template
and Create Project
Select a folder where you want to store your SIML project and Save

A SIML project file is created with the name you input (Test Create in the example)
A folder with the same name as the file is also created
Path of a valid SIML project Folder will be the Bot Directory Path you need to input in the Synbot plugin configuration form.
Close your SIML Project in Syn Bot Studio : File CloseProject
In order to install the example SIML project , download the Zip file specified in the plugin Installation Guide www.ez-robot.com/EZ-Builder/Plugins/view/167
Unzip the file in your SIML project folder.
Example SIML files will replace and overwrite the initial standard SIML files of your project.
You can now reopen your SIML Project in Syn Bot Studio - File - Recent Projects and your SIML project

Step 3 : Install , configure and use the Synbot Plugin in EZ Builder

Download and install the Synbot Plugin in EZ Builder.
www.ez-robot.com/EZ-Builder/Plugins/Category/Artificial-Intelligence
Open in EZ Cloud Appstore the Project **JD for SynBot**
Add the plugin to your EZ Builder Project (Project - Add - Plugins - Synbot Plugin)

You can start a Bot dialog session with a specific userid.
As we will see later , contextual informations can be saved for each userid and will be restored whenever we start a new session for this userid.
But before starting the first bot session we need to go to the plugin configuration form.

Select the Bot Directory Path - your SIML Project Folder
Select a Bot save Directory Path - where will be saved contextual informations global for the Bot and for each userid
Select your default language in the ComboBox
You can also define a Default UserID which will be used when no userid is input at the Bot session start.
Once OK , Save your configuration options (Save Button)
Other configuration parameters will be explained later.

Step 4 : Manage the dialog with the Bot

Starting a Bot session
A bot session can be started :
. By clicking the Start Bot Button on the plugin Main Window
Bot session is started with the userid input in the userid field or with the default userid set in the plugin configuration form if the userid field is empty.
. In EZ Scripts
```
Botstart("userid")
```
where userid is the userid for the bot session
If userid is not entered Botstart() - the default userid set in the plugin configuration form will be used
. Or for use in Blockly
```
$Botuser= "userid"
ControlCommand("Synbot Plugin", Start)
```

Once the Bot started , you have access to :
. Input Query Text Field with the Send Button in order to enter and send a text query to the bot.
. Stop Button in order to stop the session with the bot
. Save Context Button in order to save the session Context for the userid - see next in step 5

It can be only started when Bot Twitter stream is set in configuration form by sending from a twitter account a private message to the Bot twitter account.
Content of private message to start the Bot is the value of Bot Start Message which can be set in plugin Configuration Form.
Console is displaying user messages sent to the Bot and responses received from the Bot
User name linked to the session userid and Bot Name are used for display message prefixing.
Here "Jean Luc" for User Name and "Maya" for Bot name
You can change Bot name in setting Synbot BOT variable Name to the value you wanted.
And so for User Name in setting Synbot USER variable Name to the value you want.
(See later in Step 5 for working with Synbot variables)
The EZ Builder variable $Botstarted is set to 0 when there's no bot session started and to 1 when bot session is started.

Querying the Bot with a text message
Once a bot session is started , you can query the Bot with text messages.
This text messages will be analysed by the Bot Engine depending on the SIML files of the Project and a message Bot response will be returned.
There are several ways to query the Bot :
. Enter the query message in the plugin Query Text Field and press Send Button or Return Key
. Send the query message as a private message from a twitter account to the bot twitter account
. In EZ Builder scripts :

```
Botquery("query message")
```

```
Or
```
$botusermessage= "query message"
ControlCommand("Synbot Plugin", Query)
```

Before sending the query message to the SYN Bot framework - some filtering is done by the plugin to suppress unwanted characters (, ; ! ? CR/LF multiple contigous spaces )
If language used is english , possessive apostrophs if any in the message are transformed ("The Mary's dress" is transformed to "The dress of Mary")
This transformation can be unchecked in plugin configuration form.
Is limited to One word before and after contraction - For example "Blue Bird's feather" will be incorrectly transformed in "blue feather of bird" and not in "feather of blue bird"
"Bird's blue feather" will also be incorrectly transformed in "blue of bird feather" instead of "blue feather of bird" .....
Bot response message is splitted in 2 parts :
. A text message which can be (depending on plugin configuration options) speaked or not and sent to PC or EZB speakers ( EZB Command Say or SayEzb will be invoked by the plugin)
. A Command message containing EZB commands which will be invoked by the plugin
Text message Bot Response can be stored in an EZ Builder Variable which name can be set and changed in plugin configuration form.

In the above Example :
The query message was "move forward at full speed"
Text Bot response message : 'I move forward"
EZB commands invoked : SetSpeed(255 , 255) then Forward()
the "|" character is just a CR/LF separator and is changed to CR/LF (As in an EZ Buider Script) when invoking the script.

**Formatting the Bot response message in SIML files**

When the query message match a Pattern defined in a Model Block , Bot response is created in the Response Block.
Response block result is sent back to the plugin.
Only the last response (in case of [GOTO] redirections) is sent back to the plugin.
To avoid lost of multiple responses due to use of [GOTO] tag ,
SIML files Bot responses must be set as value of a USER variable named
bot_event_response

```
[User xml:space="preserve" Think:Set="bot_event_response"]DURATION CANNOT BE SET FOR THIS MOVEMENT]/User>]
```

If a response is returned by the tag, this response will be also sent back to the plugin.
The query result will be the concatenation of this response and of the value of the bot_event_response USER variable.
This variable is reinitit for each new query.
In order to be able to include EZ Commands in the Bot response, Response format must be coded in SIML as follow :

Text message§Script 1§Script 2§

or

Text message§Script 1§

the § character is only a separator.
A script is a sequence of EZ Buider commands as in an EZ Builder script , each command
```

separated by the | character - equivalent to the CR/LF in an EZ Builder script.
2 scripts can be specified and invoked asynchronously in the plugin (2 executors are used)
in the example above , the Bot response message was formatted in SIML :
I move forward§SetSpeed(255 , 255)|Forward()§
the SIML code

```
[Model]
[Pattern]move forward at full speed[/Pattern]
[Response]
I move forward§SetSpeed(255 , 255)|Forward()§
[/Response]
[/Model]
```

or better

```
Model]
[Pattern]move forward at full speed[/Pattern]
[Response]
[User Think Set="bot_event_response"]I move forward§SetSpeed(255 , 255)|Forward()§[/User]

[/Response]
[/Model]
```

Stopping the Bot session
The bot session can be stopped :
. By clicking the Stop Bot Button on the plugin Main Window
. By sending a private message from a twitter account to the bot twitter account.
Content of private message to stop the Bot is the value of Bot stop Message which can be set in plugin Configuration Form.
. In EZ Scripts :

```
Botstop()
```

Or

```
ControlCommand("Synbot Plugin", Stop)
```

Context of BOT variables values and USER variable values for the userid is saved.
Example of linking to Bing Speech RecognitionControl
In you want that the Bing Speech Results will be used for querying the Bot , configure the "script to execute for each phrase detected " in Bing Speech Recognition plugin configuration form.

```
if (!$botstarted)
ControlCommand("Synbot Plugin", Start)
sleep(1000)
endif
```

```
$botusermessage = $BingSpeech
ControlCommand("Synbot Plugin", Query)
```

What is your temperature is recognized

Using Twitter
You can start stop and send query to the bot from a twitter account in sending private messages to the bot twitter account.
Bot Twitter account must be configured and authorized in configuration form.

Step 5 : Working with Synbot SIML variables

In SIML files , you can get , set and test the value of variables.
3 types of variables can be used :

1. Global to the Bot : BOT variables
2. Specific to a userid : USER variables
3. Temporary for a user query : VAR variables

All USER variables values are saved in a file either with the Save context Button during a session or at the end of the bot session for a specific user (Stop Button of the plugin)
They are restored from the file saved values when you start a new session with the same userid.
When no saved file is existing for the user , the context is restored from the User-Settings.Siml file of your active SIML project(Same initial context for every userid)
In the same manner , All BOT variables values are also saved and restored when you start a new session with any userid.
When no saved file is existing ,the context is restored from the Bot-Settings.Siml file of your active SIML project.
In Plugin Configuration Form , you need to configure folder where saving files will be located - Bot Save Directory Path
Saving File names are automatically generated : and prefixed by userid for the session - userid-settings.siml
BE CAREFUL - If you modify content of Bot-settings or User-settings SIML files in your SIML project (with Syn Bot studio) these modifications are not applied automatically to the saved files.
If you want so , you need to report modifications in the saved files
Setting the value of a variable :
During a bot session , variables values can be set either in SIML Bot files or in EZ Builder scripts.
Variable value type is always a string type.
If the variable doesn't exit , it is created.
You can set several values (A list) for the same variable.
In SIML :
the tags Bot , User and Var are used to get or set the value of variables in the block Response.
for example :

```
[User Set="Userlanguage"]fr[/User]
```

the USER variable Userlanguage is set to value "fr"
If you want to add a list of value to a variable - replace the SET keyword by ADD
In EZ Builder Scripts :

```
Setbotvar("variabletype:variablename" , "value")
```

where variabletype equal var ou user or bot
For example : Setbotvar("user:Userlanguage" , "fr")
or for using in Blockly

```
$Botvarname = "user:Userlanguage"
$Botvarvalue = "fr"
ControlCommand("Synbot Plugin", Setvar)
```

```

**Getting  the value of a variable :**
In SIML :
the tags Bot  , User  and Var  are used  to get or set the value of variables in the  block Response.
for example :
```
[User get="Userlanguage" /]
```
 return the value of the USER variable Userlanguage
If several values are set for the same Bot variable , only the first one is returned  with GET ....
If you want to get the  the  list of values for a variable - replace the GET keyword by LIST
The value of the BOT variable "List-Separator"  is used to configure the separator between every value of the list.
In EZ Builder Scripts :
Gettbotvar("variabletype:variablename" )
For example :
```
Getbotvar("user:Userlanguage")
```

or for using in Blockly

```
$Botvarname = "user:Userlanguage"
ControlCommand("Synbot Plugin", Getvar)
```

Result value is returned in $Botvarvalue EZB variable
If several values are set for the same Bot variable , only the first one is returned ....List is not supported in this version

Testing the value of a variable
In SIML :
use the [If] or [Switch] tags
For example

```
[If User="Userlanguage" Value="fr"][Goto xml:space="preserve"]XXXXXXINTERPRETATIONOFMOVEMENTFLAGS [Match Index="2" /][/Goto][/If]
```

In EZ Builder Scripts :
Get the value of the variable
Use logic operators with the result returned in $Botvarvalue EZB variable
For complementary informations - look at SIML tutorial below
developer.syn.co.in/tutorial/siml/elements/bot.html
https://developer.syn.co.in/tutorial/siml/elements/usersettings.html
https://developer.syn.co.in/tutorial/siml/elements/variable.html
```

Step 6 : Working with SIML Events

SIML Events allows to trigger special evaluation of responses in SIML files
In SIML files you define an event with the EVENT tag

```
 [Event]
[Pattern]TEST-EVENT[/Pattern]
[Response]Event Raised.[/Response]
[/Event]
```

In SIML files , you can trigger the event TEST-EVENT with the RAISE tag :

```
[Model]
[Pattern]RAISE EVENT[/Pattern]
[Response]
[Raise]test-event[/Raise]
[/Response]
[/Model]
```

Response of the bot to user message "Raise event" will be "Event Raised"
Response of the bot will be sent and managed by the plugin as the response to a normal query

Raising SIML events in EZ Builder scripts

You can raise SIML events in EZ Builder scripts :

```
Raisebotevent("bot event id")
```

Or

```
$Botevent= "bot event id"
ControlCommand("Synbot Plugin", Raiseevent)
```

Where bot event id is the pattern of the SIML event

```
Raisebotevent("test-event")
```

Will raise the test-event event.
Response of the bot will be "Event Raised"
See for example User Input recognition for "What do you See" which call an EZBuilder Script with use of Cognitive Vision Plugin.

```
[Case Value="CAMERA_RECOGNITION"][Var Set="EZscript"]Start Bing Vision Recognition[/Var][Var Set="mess1"]|Waiting for Bing Vision recognition[/Var][/Case]
```

```

Synbot Event is set in this script .

```
if (!$IsCameraActive)
ControlCommand("Camera", CameraStart)
sleep(1000)
endif
ControlCommand("Cognitive Vision", Detect)
sleep(500)
$botevent = "Return from Bing Vision Event"
ControlCommand("Synbot Plugin", Raiseevent)
```

Model is coded for this event in order to process Cognitive vision Plugin answer (especially translate it for french language).

```
[Event]
[Pattern]
[!--Event set by EZscript Script in script Manager Start Bing Vision Recognition after execution of Bing Vision recognition
Only in english - so need to be modified to include the [x:Translate] which is in testing
Need to be also modified if Bing result are Empty
--]
[Item]RETURN FROM BING VISION EVENT[/Item]
[/Pattern]
[Response xml:space="preserve"]
[Think]
[Var Set="Visiondescription"][x:EZvar Get="VisionDescription" /][/Var]
[Var Set="Visionconfidence"][x:EZvar Get="VisionConfidence" /][/Var]
[Var Set="Visiondescriptionfr"][x:Translate][Var Get="Visiondescription" /][/x:Translate][/Var]
[If Var="Visionconfidence" Not="0"]
[Switch User="Userlanguage"]
[Case Value="fr"]
[Var Set="Visiondescription"][x:Translate][Var Get="Visiondescription" /][/x:Translate][/Var]
[Var Set="Temp"] Je vois [/Var][Var Set="Temp1"]avec une certitude de [/Var][Var Set="Temp2"] pourcent[/Var][/Case]
[Case Value="en"]
[Var Set="Temp"]I See [/Var][Var Set="Temp1"]with a level of confidence of [/Var][Var Set="Temp2"] percent[/Var][/Case]
[/Switch]
[Var Set="Messresponse"][Var Get="Temp" /][Var Get="Visiondescription" /][Var Get="Temp1" /]_[Var Get="Visionconfidence" /][Var Get="Temp2" /][/Var]
[/If]
[Else]
[Switch User="Userlanguage"]
[Case Value="fr"][Var Set="Messresponse"]Je ne peux rien voir[/Var][/Case]
[Case Value="en"][Var Set="Messresponse"]I cannot see anything[/Var][/Case]
[/Switch]
[/Else]
[/Think]
```
```

```
[Var Get="Messresponse" /]
[/Response]
[/Event]
```

Below the result ....

Step 7 : Accessing in SIML to EZ Builder variables and EZ Builder Command

Invoke the execution of EZB Command In SIML
[x:EZcmd] tag
With [x:EZcmd] tag , you can execute an EZ Builder command
```
[x:EZcmd]ControlCommand("Speech Settings", SetVoice, "CereVoice Suzanne - French [France]")[/x:EZcmd]
```

A sequence of EZB commands can be invoked as in an EZB script.
Each command is separated by the | separator which is interpreted as CR/LF.
```
[[x:EZcmd]ControlCommand("Speech Settings", SetVoiceEmphasis, "Reduced")|ControlCommand("Speech Settings", SetVoiceRate, "Medium")[/x:EZcmd]
```

if the command return a result this result is returned,
for exemple
```
 [x:EZcmd]Print(GetVoltage())[/x:EZcmd]
```
returns the EZB voltage
You can easily with EZcmd tag set value of an EZ Builder Variable.

```
[x:EZcmd]$AwaitforSIML = 1[/x:EZcmd]
```

Access to EZ Builder variables values In SIML
[x:EZvar] tag
With [x:EZvar] tag , you can get the value of an EZ Builder variable or an empty string if the variable does not exist.
Name of EZ builder variable is specified without the first $ character.

```
[x:EZvar Get="Direction" /]
```

you get the value of the EZ Builder variable $Direction
With [x:EZvar] tag , you can also set the value of an EZ Builder variable - if the variable is not existing , it is created

```
[x:EZvar Set="AwaitforSIML" Value="1" /]
```

You can so synchronize an EZB Builder script with the setting in SIML of a variable.

```
[Model]
[Pattern]READY TO GO FORWARD ROBOT[/Pattern]
[Response]
```

[User Think:Set="Flagforsynchro"]1[/User]
[x:EZvar Set="AwaitforSIML" Value="0" /]
I am waiting for your yes go guy§ControlCommand("Script Manager", ScriptStart, "Synchro forward SIML")§[/Response]
[/Model]

[Model]
[Pattern]YES GO GUY[/Pattern]
[Response]
[If User="Flagforsynchro" Value="1"]
[x:EZvar Set="AwaitforSIML" Value="1" /]
I Will go Forward
[User Think:Set="Flagforsynchro"][/User][/If]
[/Response]
[/Model]
```

The EZB script Synchro forward SIML is defined as :

```
WaitFor($AwaitforSIML = "1")
Forward(140)
```

and next :

SIML User variable Flagforsynchro is used only to be sure than we are awaiting really the YES GO GUY.
Getting or setting of Array EZB Variables is not supported in this Plugin version
```

Step 7 bis : Working with Bot Emotions

Emotions can be configured in SIML File EMOTIONML in Settings Folder.
Setting the value of Bot Emotion :
During a bot session , Bot emotion can be set either in SIML Bot files or in EZ Builder scripts.
Variable value type is always a string type.
If the variable doesn't exit , it is created.
In SIML : **[BotEmotion ID="Sad" /]**
In EZ Builder Scripts :

```
Setbotemotion("Emotion ID")
```

or for using in Blockly

```
$Botemotion = "Emotion ID"
ControlCommand("Synbot Plugin", Setbotemotion)
```

Getting the value of Bot Emotion :
In SIML : **[BotEmotion Get="id" /]**
In EZ Builder Scripts :

```
Gettbotemotion()
```

or for using in Blockly

```
ControlCommand("Synbot Plugin", Getbotemotion)
```

Result value is returned in $Botemotion EZB variable
Synbot Plugin Configuration:
Whenever Bot Emotion value is changed , value of $Botemotion EZB variable is updated.
You can also in Plugin Configuration Form specify a default init value for Bot Emotion which will be set when you start the Bot Session.
if the configured default Emotion is Empty , Bot Emotion will be saved and restore from one session to another for the same User , if the configured default Emotion is Empty.
You can also configure an EZ Builder Script which will be executed every time Bot emotion Changed.

Framework integration:
In order to demonstrate use of Bot Emotion , framework example include some new features
A new [Verb] Show Yourself in order to be able to set Bot Emotion - which is a [free attribute]
Verb bbId and Verb AuthoId are set to "EMOTION"
Map Emotion is used to control free attribute (3rd word in content is link to the emotion ID)

```
[Map Name="Emotion"]
[MapItem Content="fr|triste" Value="%1M EMPTY sad" /]
[MapItem Content="en|sad" Value="%1M EMPTY sad" /]
[MapItem Content="fr|en colère" Value="%1M EMPTY angry" /]
[MapItem Content="fr|furieux" Value="%1M EMPTY angry" /]
[MapItem Content="fr|furax" Value="%1M EMPTY angry" /]
[MapItem Content="fr|en rage" Value="%1M EMPTY angry" /]
```

and Map Emotion_lib is used to retrieve a label corresponding to Emotion ID and language used

```
[Map Name="Emotion_lib"]
[MapItem Content="fr|0|sad" Value="triste" /]
[MapItem Content="en|0|sad" Value="sad" /]
[MapItem Content="fr|0|angry" Value="en colère" /]
[MapItem Content="en|0|angry" Value="angry" /]
[MapItem Content="fr|0|bored" Value="ennuyé" /]
[MapItem Content="en|0|bored" Value="bored" /]
```

Action for EMOTION Workflag has been created with lauching of an EZBuilder Script

```
[Case Value="EMOTION"][Goto xml:space="preserve"]XXXXXXXCHANGEEMOTION[/Goto]
[Var Set="EZscript"]Bot Emotion Change Action[/Var][/Case]
```

code] [Model]
[Pattern]XXXXXXXCHANGEEMOTION[/Pattern]
[Response xml:space="preserve"]
[Think]
[Var Set="Messchangedemotion"]Messchangedemotion_[User Get="Userlanguage" /][/Var]
[Var Set="emotionname"][Text WordAt="3"][User Get="Contentmapfree" /][/Text][/Var]
[!--Change Bot Emotion--]
[BotEmotion ID="" /]
[Var Set="Messchangedemotion"]Messchangedemotion_[Var Get="emotionname" /]*[User Get="Userlanguage" /][/Var]*
[!--Test if a specific message exist for the bot emotion id--]
[Var Set="Emotionanswerexisting"][Random Get="" /][/Var]
[!--If not take the default message--]
[If Var="Emotionanswerexisting" Value=""]
[Var Set="Messchangedemotion"]Messchangedemotion[User Get="Userlanguage" /][/Var]
[/If]
[!--Message will be said with waiting--]
[Var Set="Mess1replace" xml:space="preserve"]¤[Random Get="" /][/Var]
[/Think]
[/Response]
[/Model][/code]
Some news models have been integrated in Bot Dialogs SIML file , in order to be able to query on Bot Emotion.

Step 7 Ter: Bot Save Directory and Learning Models

Bot Save Directory
the Bot save Directory Path can be configured in Plugin Configuration Form.
Some files will be created ans saved with every bot cession in this Folder :
Bot-Settings.siml
Values of Bot variables will be saved in this file at the end of Bot session (Stop Bot) or when Save Button is pressed
[Userid]-Settings.siml where [Userid] is a Bot session Userid
Values of User variables will be saved in this file at the end of Bot session (Stop Bot) or when Save Button is pressed
When starting a new Bot session, values of Variables saved in this files will be automatically restored
Learned.siml
Learn Models (See SIML tag LEARN) will be saved in this file in append mode when learned.
[Userid]-Memorized.siml
Memorized Models for the UserID (See SIML tag REMEMBER) will be saved in append mode in this file when learned
When starting a new Bot session, Modes saved in this files will be automatically loaded and activated (depending on User Id for the memorized models which
are memorized only for a User Id)
You can in this Bot save folder create some subfolders :
Learned and [Userid]-Memorized (for example if one of your User ID is John you can create John-Memorized folder).
Every Files in these folders must me siml files.
Models included in these files will be automatically load and activated when starting a Bot session(depending on User Id for the files in UserId-Memorized folders which
will be load only if the session UserId is the same)
Learn SIML tag usage example
TheLearn tag is used to save a Model into the Bot's Knowledge Base and this unit of knowledge will then be available to all users.
Suppose you wish to teach your Bot that John live in Boston and Carol live in Paris
```

[Siml xmlns:x="https://syn.co.in/2014/siml#external" xmlns:Think="https://syn.co.in/2014/siml#think"]
[Concept Name="Learn" Type="Public"]

```
[Model]
  [Pattern]* LIVE IN *[/Pattern]
  [Response xml:space="preserve"]
   [User xml:space="preserve" Think:Set="bot_event_response"]Alright I will keep in mind that [Match /] is living in [Match At="2" /][/User]
   [Learn]
```

[Model]
[Pattern]
[Item]WHERE IS LIVING [Process][Match /][/Process][/Item]
[Item]XXXXXFROMPATTERNREDUCTION1 WHERE IS LIVING [Process][Match /][/Process][/Item]
[/Pattern]
[Response xml:space="preserve"]
[Think]
[User xml:space="preserve" Set="bot_event_response"][Process][Match /][/Process] is living in [Process][Match At="2" /][/Process][/User]
[/Think]
```

[/Response]
[/Model]

```
[/Learn]
  [/Response]
[/Model]
```

[/Concept]
[/Siml]
```

In file Learned.Siml when you Input "John live in Boston" , the following Model will be generated
```
[Siml xmlns:x="https://syn.co.in/2014/siml#external" xmlns:Think="https://syn.co.in/2014/siml#think"]
[Concept Name="Learn" Type="Public" Repeat="true"]

```
[Model]
  [Pattern]
    [Item]WHERE IS LIVING JOHN[/Item]
    [Item]XXXXXFROMPATTERNREDUCTION1 WHERE IS LIVING JOHN[/Item]
  [/Pattern]
  [Response xml:space="preserve"]
    [Think]
     [User xml:space="preserve" Set="bot_event_response"]john is living in boston[/User]
     [/Think]
  [/Response]
[/Model]
```

[/Concept]
[/Siml]
```

And so Bot response to "Where is living John User Input" will be correct

I Recommend that you create empty Learned.siml and [userid]-Memorized.siml initialized with
```
[Siml xmlns:x="https://syn.co.in/2014/siml#external" xmlns:Think="https://syn.co.in/2014/siml#think"]
[/Siml]
```
```

Step 8 : Advanced Plugin Configuration options

Plugin configuration form is only available when no Bot session active (Bot stopped)

Link to EZ Scripts
You can configure in **Response Variable** field the EZ Builder variable name where the text part of the Bot Response will be stored - default is $SynbotTextResponse
You can configure EZ Builder Scripts to execute :
. When the plugin received a Bot Response (Field **Response Script**)
. Just after a Bot session start (Field **Start Script**)
. Just before a bot session stop (Field **Stop Script**) - BE CAREFUL This script cannot include Bot Commands or Bot Variables Getting or settings
. When Bot Emotion value is changed
Bot response interpretating options
Text Part of the Bot Response if existing will send automatically a SAY or a sayEZB command to EZ Builder with nowait.
you can configure the following options for the text message interpretation :.
If **box speak response** is unchecked , No SAY commands are executed
. If box speak response is ckecked
. Case **box Use EZB-V4 speaker** is ckecked , SayEZB("text message") is executed
. if not checked - Say("text message") is executed
If text message begin with | character the message is only displayed on Bot Console , whenever the speak flags settings
If text message begin with ¤ character the message is speaked and in case of SayEZB , execution of EZ Command is waiting for End of speaking
If **box execute response command** is checked , Command part of Bot response will be launched in execution by EZ Builder.
If not checked , Command part is only displayed on the Console.
Default is checked.
box Log Full bot Response is only for debug (so get unchecked)
box Authorize Batch entries If checked , you will be able to submit sequences of Bot query user inputs stored in external text files.
Text files must be located in the Bot Save Directory folder.
A new field will be showed in the Plugin main window after Bot starting in order to input Batch file name to use.
The EXEC button when pressed allow to launch sequential reading of Quey user inputs stored in the Batch text file.
This option can be used for unit testing and bot learning automation.

box Log Bot Dialog if checked bot dialog as displayed in bot console will be recorded in a file logdialog.txt located in the Bot Save Directory folder.
Each Bot session (between Start Bot and Stop Bot) is timestamped - bot Dialog for a session is recorded in append mode to the mog file.
This option combined with the Authorize Batch Entries allow easily to automate Bot Unit testing.
Possessive Apostroph filter if checked , if default Language is En(English) , if user input include possessive apostroph form extension will be done before sending the message to bot analysis - "the John's car" will be transformed on "the car of John" , making easier the SIML models.
Learning mode If checked learning mode will be activated (Planned in next version)
Default Language
When a bot session is started , First part of the Combo Box value (The language code) is stored in the Bot USER variable "Userlanguage" and can be used and tested in SIML files

Robot Type
When a bot session is started , field value is stored in the Bot USER variable "Robottype" and can be used and tested in SIML files
Default Bot Emotion
Default init value for Bot Emotion which will be set when you start the Bot Session.
if the configured default Emotion is Empty , Bot Emotion will be saved and restore from one session to another for the same User , if the configured default Emotion is Empty.
Init Bot event
You can configure a Bot Event which will be raised after starting a Bot session.
In this Bot event, you can do some initializations for the session (for example Bot and User variables value settings)
Default is BOT INIT EVENT
Below is the SIML model used in the example files to manage this init Bot event.
Speech settings EZB Control must be active in order to set the voice parameters.

```
[Event]
[Pattern]BOT INIT EVENT[/Pattern]
[Response xml:space="preserve"]
[Think]

[Bot Set="Copylearnto_en"]False[/Bot]
[Bot Set="Copylearnto_fr"]True[/Bot]
[User Set="Textentered"][/User]
[Bot Set="Deltaspeed_leftright"]30[/Bot]
[User Set="Workspeedleft"]100[/User]
[User Set="Workspeedright"]100[/User]
[Bot Set="EZmediafolder"]C:\Users\benar\OneDrive\Images\My Robot Pictures[/Bot]
[User Set="EZMediafolder"][x:EZinfo Get="cameramediasavefolder" /][/User]
[If User="EZMediafolder" Value=""][User Set="EZMediafolder"][Bot Get="EZmediafolder" /]
[/User][/If]
[Bot Set="Listseparator_end_fr"] et [/Bot][Bot Set="Listseparator_end_en"] and [/Bot]
[Bot Set="Listseparator_1_fr"] mais aussi [/Bot][Bot Set="Listseparator_1_en"] but also [/
Bot]
[Bot Set="Listseparator_2_fr"] et aussi [/Bot][Bot Set="Listseparator_2_en"] and finally [/
Bot]
[Var Set="Controlrobottype"][Map Get="typerobot"][User Get="Robottype" /][/Map][/Var]
[!--test if Controlrobottype = RobotType in this case the robottype is not existing in the bot
robot table - no control  depending on Robottype will be done after --]
[!--User variable indrobot is set to the index to control in the flag arrays - if empty no
control will be done--]
[If Var="Controlrobottype" Value=""][User Set="indrobot"][/User][/If]
[!--[Var Set="Okrobot"][x:Compare]%equal%[/x:Compare][/Var]
[If Var="Okrobot" Value="true"][User Set="indrobot"][/User][/If]--]
[Else][User Set="indrobot"][Var Get="Controlrobottype" /][/User][/Else]
[Switch User="Userlanguage"]
[Case Value="fr"][x:EZcmd]ControlCommand("Speech Settings", SetVoice, "CereVoice
Suzanne - French [France]")[/x:EZcmd][/Case]
[Case Value="en"][x:EZcmd]ControlCommand("Speech Settings", SetVoice, "Microsoft Zira
Desktop")[/x:EZcmd][/Case]
[/Switch]
[x:EZcmd]ControlCommand("Speech Settings", SetVoiceEmphasis, "Reduced")|
ControlCommand("Speech Settings", SetVoiceRate, "Medium")[/x:EZcmd]
[Switch User="Robottype"]
[Case Value="JD"][Bot Set="robottypefr"]un robot humanoïde[/Bot][Bot
```

Set="robottypeen"]an humanoid robot[/Bot][/Case]
[Case Value="ROLLI"][Bot Set="robottypefr"]un robot à chenilles[/Bot][Bot Set="robottypeen"]a rolling rover robot[/Bot][/Case]
[/Switch]
[Bot Set="Empty-Response"]
[Switch User="Userlanguage"]
[Case Value="fr"][Random Get="Sorry_fr" /] JE NE SAIS PAS RéPONDRE à ta demande[/Case]
[Case Value="en"][Random Get="Sorry_en" /] I Don't understand your request[/Case]
[/Switch]
[/Bot]
[Bot Set="Timeout-Response"]
[Switch User="Userlanguage"]
[Case Value="fr"][Random Get="Sorry_fr" /][Random] Le temps nécessaire pour traiter ta demande est trop long[/Random][/Case]
[Case Value="en"][Random Get="Sorry_en" /][Random] Your request has timed out[/Random][/Case]
[/Switch]
[/Bot]
[/Think]
[/Response]
[/Event]
```

**Bot Welcome**
You can configure a user Welcome Message which will be automatically sent to the Bot after starting a Bot session and after the init Bot event management.
Default is WELCOME TO SYNBOT
If user name is not set , a dialog is initiated by the Bot asking user to input his name.
Userlanguage variable is tested to adapt bot dialogs and bot Below is the SIML models used in the example files to manage this welcome message.
responses to the user language (fr and en are only supported in the example files)

```
[Model]
[Pattern]
[Item]WELCOME TO SYNBOT[/Item]
[/Pattern]
[Response xml:space="preserve"]
[If User="Name" Value=""][Var Think:Set="Temp"]NameusernotOK[/Var][/If]
[Else][Var Think:Set="Temp"]NameuserOK[/Var][/Else]
[Goto xml:space="preserve"]WELCOME TO SYNBOT [Var Get="Temp" /] [User Get="Userlanguage" /][/Goto]
[/Response]
[/Model]
```

```
[Model]
  [Pattern]
    [Item]WELCOME TO SYNBOT NAMEUSEROK FR[/Item]
  [/Pattern]
  [Response]
    [Random Get="Welcome_fr" /]
  [/Response]
[/Model]

[Model]
  [Pattern]
```
```

```
    [Item]WELCOME TO SYNBOT NAMEUSEROK EN[/Item]
  [/Pattern]
  [Response xml:space="preserve"]
    [Random Get="Welcome_en" /]
  [/Response]
[/Model]

[Model]
  [Pattern]
    [Item]WELCOME TO SYNBOT NAMEUSERNOTOK FR[/Item]
  [/Pattern]
  [Response xml:space="preserve"]
    [Label]:Setusername-fr[/Label]
    [User Think:Set="bot_event_response"][Random Get="Bienvenue_mess_fr" /]_[Random
Get="Presentation_Name_fr_1" /] je suis [Bot Get="robottypefr" /] MAIS [Random
Get="Je_ne_connais_pas_ton_nom_fr" /][/User]
    [/Response]
[/Model]

[Model]
  [Pattern]
    [Item]WELCOME TO SYNBOT NAMEUSERNOTOK EN[/Item]
  [/Pattern]
  [Response xml:space="preserve"]
    [Label]:Setusername-en[/Label]
    [User Think:Set="bot_event_response"][Random Get="Bienvenue_mess_en" /]_[Random
Get="Presentation_Name_en_1" /] I'm [Bot Get="robottypeen" /] but [Random
Get="Je_ne_connais_pas_ton_nom_en" /][/User]
    [/Response]
[/Model]
[Model]
  [Pattern]
    [Item]JE M'APPELLE *~2[/Item]
    [Item]TU PEUX M'APPELER *~2[/Item]
    [Item]MON NOM EST *~2[/Item]
    [Item]MON PRÉNOM EST *~2[/Item]
    [Item]APPELLE MOI *~2[/Item]
    [Item]*~2[/Item]
  [/Pattern]
  [Previous]:Setusername-fr[/Previous]
  [Response xml:space="preserve"]
    [User Think:Set="Name"][Match /][/User]
    [Random]
      [Item][Random Get="Presentation_response_deb_fr_1" /]_[Random
Get="Presentation_response_fin_fr" /][/Item]
      [Item][Random Get="Presentation_response_deb_fr" /] Je T'APPELERAI MAINTENANT [Match /]
[/Item]
      [Item][Random Get="Presentation_response_deb_fr" /] Je T'APPELERAI [Match /] à partir
de maintenant[/Item]
    [/Random]
  [/Response]
[/Model]
[Model]
  [Pattern]
    [Item]MY NAME IS *~2[/Item]
    [Item]YOU CAN CALL ME *~2[/Item]
    [Item]CALL ME *~2[/Item]
    [Item]*~2[/Item]
  [/Pattern]
  [Previous]:Setusername-en[/Previous]
  [Response xml:space="preserve"]
    [User Think:Set="Name"][Match /][/User]
    [Random]
      [Item][Random Get="Presentation_response_deb_en_1" /]_[Random
Get="Presentation_response_fin_en" /][/Item]
      [Item][Random Get="Presentation_response_deb_en" /] I will call you [Match /][/Item]
      [Item][Random Get="Presentation_response_deb_en" /] I will call you [Match /] now[/
Item]
    [/Random]
  [/Response]
```

```
[/Model]
```

For example for a new userid "John"

Before Stop Bot event

You can configure a Bot Event which will be raised just after the Bot stop Command and just before the session finished.
Default is BOT STOP EVENT
You can do in the model used for this event some savings for the session (for example Bot and User variables value settings)
Below is the SIML model used in the example files to manage the Before Stop Event in sending to the user a Random Bye Message Response customized depending on the User language used.

```
[Event]
[Pattern]
[Item]BOT STOP EVENT[/Item]
[/Pattern]
[Response xml:space="preserve"]
[User Think:Set="EZMediafolder"][/User]
[Random Get="Bye_Bye_" /]
[/Response]
[/Event]
```

```
[Random Name="Bye_Bye_fr"]
[Item xml:space="preserve"]AU REVOIR [User Get="Name" /] A BIENTOT[/Item]
[Item xml:space="preserve"]AU REVOIR et A BIENTOT [User Get="Name" /][/Item]
[Item xml:space="preserve"][Random Get="Bye_bye_part1_fr" /]
[Random Get="Bye_bye_part2_fr" /][/Item]
[Item xml:space="preserve"][Random Get="Bye_bye_part1_fr" /]
[Random Get="Bye_bye_part2_fr" /]
[User Get="Name" /][/Item]
[/Random]
[Random Name="Bye_bye_part1_fr"](AU PLAISIR de te revoir bientot |REVIENS VITE me voir)[/Random]
[Random Name="Bye_bye_part2_fr"](tu me manques déjà|
)[/Random]
[Random Name="Bye_Bye_en"]
[Item xml:space="preserve"][Random Get="Bye_end_en" /]
[User Get="Name" /]
[Random Get="See_You_next" /][/Item]
[Item xml:space="preserve"][Random Get="Bye_end_en" /]
[Random Get="See_You_next" /]
[User Get="Name" /][/Item]
[Item xml:space="preserve"][User Get="Name" /]
[Random Get="Bye_end_en" /]
[Random Get="See_You_next" /][/Item]
[/Random]
```

[Random Name="Bye_end_en"](Bye|Bye Bye|Pleased to meet you|Glad to meet you|It was a pleasure)[/Random]
[Random Name="See_You_next"]
[Item xml:space="preserve"]See You [Random Get="Bye_bye_part3_en" /][/Item]
[Item xml:space="preserve"]I Miss You[/Item]
[Item xml:space="preserve"]I Miss You already[/Item]
[/Random]
[Random Name="Bye_bye_part3_en"](next|soon|later|another time|_)[/Random]
```

**Bot no match Message**

You can configure a Message which will be automatically sent to the Bot when the plugin receive from the Bot an Empty answer Response.
Default is BOT NO RESPONSE MESSAGE.
If there are no Match Patterns for a user query , Bot variable "Empty-Response" can be set to a message value.
The message value Set will be returned as Response to the User.
This Bot mechanism is also activated when the last response to a user request is Empty - It doesn't mean Response of the last model activated but Response of the first Model activated in the query ( It can be confusing with several GOTO ....)
So you can decide of mechanism to use in setting or not a response in the last model activated in a GOTO chaining.
The Plugin will consider that there's No Match when normal Bot Response message (Response of the first model activated) is empty and when variable used to store the Bot Response (User variable bot_event_response) is empty.
In this case, if a Bot No Match Message is configured , it is send as querying request to the Bot.
This mechanism allow to configure in the Bot no Match Message Model a customized message to be sent to the User in case of a Bot Empty Response (Generally due to an error in user input)
Below is the SIML models used in the example files to manage the Bot No Response Message with customization depending on the User language used.

```
[Model]
[Pattern]
[Item]BOT NO RESPONSE MESSAGE[/Item]
[/Pattern]
[Response xml:space="preserve"]
[Think]
[Var Set="Can_not_answer"]Can_not_answer_[User Get="Userlanguage" /][/Var]
[Var Set="Sorry"]Sorry_[User Get="Userlanguage" /][/Var]

[User Think:Set="bot_event_response"]
[Random]
[Item][Random Get="" /][/Item]
[Item][Random Get="" /][User Get="Name" /][/Item]
[Item][Random Get="" /][User Get="Name" /][Random Get="" /][/Item]
[Item][Random Get="" /][Random Get="" /][/Item]
[/Random]
[/User]
[/Think]
[/Response]
[/Model]
```
```

```
[Random Name="Can_not_answer_fr"](Je ne sais pas répondre à ta demande|Je ne sais pas répondre)[/Random]
[Random Name="Can_not_answer_en"](I can't respond to your request|I can't answer to your request|I can't answer|I can't respond)[/Random]
[Random Name="Sorry_fr"](Je suis désolé|Désolé|_)[/Random]
[Random Name="Sorry_en"](I'm sorry|Sorry| )[/Random]
```

Bot start and Stop Message
You can configure which text message will be used in an input use message , to start and stop the Bot. Useful for example in case of input done
by the way of private twitter messages.
Default are "Start Bot" and "Stop Bot"
Advanced Tab configuration Parameters
When you click on advanced tab, you get some additionnal configuration parameters.

Bot Email Configuration
In order to be able to use the Mail SIML tag, you need to configure Bot/Robot Email parameters : Email address , Email Password , SMTP Server Address SMTP Server Port.
API Translate Configuration
In order to be able to use the Translate SIML tag, you need to input yout Text Translate API Key : Assigned by Microsoft in Azure when the Text_Translate service is created for your Account.
Bot Twitter Configuration
In order to be able to query the bot with Twitter private Messages and to use the Twitter SIML Tag , you need to Configure and Authorize Twitter Bot/Robot Account - Same process as Twitter account configuration in EZ Builder Option
But allowing Private messages - It can be and it is better to use the same Bot/Robot Twitter Account as in Ez Builder global configuration.
Check or Uncheck Tweet Bot Response Message to allow or not sending of Bot Text Response Message to the Twitter User who send the querying private message.

Step 9 : Adding new SIML Tags to the language

One of the main strength of Syn Bot Framework is the ability to code in C # custom adapters to handle new SIML Tags.
In SIML files you need to precise in the SIML tag that external adapters will be used
```
[Siml xmlns:x="https://syn.co.in/2014/siml#external" xmlns:Think="https://syn.co.in/2014/siml#think"]
```

In SIML , if we define an adapters with name **newtag** - tag Syntax will be **[x:newtag][/x:newtag>]**
Synbot Plugin include reference to a UserSIMLadaptator.dll than you can code to create classes for handling custom new tags
You need just to download and unzip the Visual studio Solution here :
UserSIMLadaptators.zip
Then to mofify in Visual Studio adding your code and generate
then to go to the BIN\DEBUG folder of the project
and to copy the files UserSIMLadaptators.dll , UserSIMLadaptators.dll.config and UserSIMLadaptators.pdb in the synbot plugin directory C:\Users\Public\Documents\EZ-Builder\Plugins(GUID of Synbot Plugin)
Here example of coding for the TextopAdapter adaptator processing SIML tag [x:Textop][/x:Textop>]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using Syn.Bot.Siml;
using Syn.Bot.Siml.Interfaces;
using Syn.Bot.Siml.Events;
using System.Diagnostics;
using System.Collections;
namespace UserSIMLadaptators
{
public class Simladaptators
{
public static bool traceEZlog = false;
private static Simladaptators instance;
static Simladaptators()
{
instance = new Simladaptators();
}
    public static Simladaptators adaptators
    {
        get { return instance; }
    }
    private Simladaptators() { }
    public void Add(string x, ref List listadapters)
    {
```

```
        // Add here to the list all new adapters you want to integrate method name as string
        listadapters.Add("Textop");
        return;
    }
    public static string Textop(Context parameter)
    {
        string function = string.Empty; string result = false.ToString(); string inputtext =
string.Empty; string delimitor = string.Empty;
        string charat = string.Empty; string length = string.Empty; string value =
string.Empty; int textlength = 0;
        int numat = 0; int numlength = 0;
        if (parameter.Element.Value != null) inputtext = parameter.Element.Value;
        if (parameter.Element.Value == null || parameter.Element.Value == string.Empty)
return string.Empty;
        textlength = inputtext.Length;
        if (parameter.Element.Attribute("Function") != null) function =
parameter.Element.Attribute("Function").Value.ToLower();
        if (parameter.Element.Attribute("Delimitor") != null) delimitor =
parameter.Element.Attribute("Delimitor").Value.ToLower();
        if (function == string.Empty) function = "substring";
        if (function == "split" && delimitor == string.Empty) return string.Empty;
        if (parameter.Element.Attribute("At") != null)
        {
            charat = parameter.Element.Attribute("At").Value.ToLower();
            if (!Int32.TryParse(charat, out numat)) numat = 0;
            else
            {
                if (numat > textlength) numat = 0;
            }
        }
        if (parameter.Element.Attribute("Length") != null)
        {
            length = parameter.Element.Attribute("Length").Value.ToLower();
            if (!Int32.TryParse(charat, out numlength)) numlength = 0;
        }
        if (parameter.Element.Attribute("Value") != null) value =
parameter.Element.Attribute("Value").Value;
        if (function == "replace" && value == string.Empty) return inputtext;
        if (function == "replace" || function == "substring")
        {
            if (numat == 0) return string.Empty;
        }
        var paruser = parameter.User;
        EZ_Builder.EZBManager.Log("USER {0}:{1}", "split", paruser);
        switch (function)
        {
            case "substring":
                {
                    if (numlength == 0) return inputtext.Substring(numat - 1);
                    if (numat + numlength > textlength + 1) return inputtext.Substring(numat
- 1);
                    return inputtext.Substring(numat - 1, numlength);
                }
            case "replace":
                {
                    if (numlength == 0) numlength = value.Length;
                    StringBuilder sb = new StringBuilder(inputtext);
                    StringBuilder sbvalue = new StringBuilder(value);
                    int j = 0;
                    for (int i = 1; i < numlength + 1; i++)
                    {
                        sb[numat + i - 2] = sbvalue[j];
                        j++;
                    }
                    return sb.ToString();
                }
            case "split":
                {
                    string[] ressplit = null;
                    ressplit = inputtext.Split(delimitor.ToCharArray());
```

```
                    for (int i = 1; i < ressplit.Length + 1; i++)
                    {
                        // Store results in SIML Var
                        parameter.User.Vars["ressplit_" + i.ToString()].Value = ressplit[i -
1];
                    }
                    return ressplit.Length.ToString();
                }
            default:
                {
                    return string.Empty;
                }
        }
    }
 }
```

}
```

You can easily in the adapter code get or set Synbot variables
for Var variables  - parameter.User.Vars["variable name"].Value
For User variables - parameter.User.Settings["variable name"].Value
For Bot variables  - parameter.Bot.Settings["variable name"].Value
```

Step 10 : New SIML Tags added in the Plugin - Wikipedia Search , Translate , Twitter , Mail , ...

WIKIPEDIA Search
[x:Wikipedia] tag
With [x:Wikipedia] tag , you can search on Wikipedia either with MediaWiki API , either Opensearch API or Query API.
Query API give more detailed results than Opensearch API.
Syntax is
[x:Wikipedia Search="Searched expression" Filter="No" /] for Opensearch API
[x:Wikipedia Search="Searched expression" Action="query" Filter="No" /] for Query API
Querying language is the default language configured in the plugin configuration form.
Number of found results is returned and can be tested.
In case of network access error - 0 is returned.
Filter parameter is an option to filter or not wikipedia response content - implicit is filter activated - if you don't want you need to specify Filter="No".
Filtering suppress all text within () in the response and entries ending with "refers to:"

```
[User Set="Tempcount"][x:Wikipedia Search="" /][/User]
```

Results found can be accessed in SIML User variables (5 results maximum are returned) - variables Result_1 Result_2 Result_3 Result_4 and Result_5
Bot querying models examples can be found in "Search Wikipedia SIML" file.
Or you can try the batch test file Testwikien.txt
Here an example of Dialog :

Translate
[x:Translate] tag
With [x:Translate] tag , you can translate a text from a source language to another language using in background the Microsoft Bing Translate API.
You need first to subscribe free to the TRANSLATE_TEXT service in your Microsoft Azure Dashboard.
Free plan include 2 million characters per month.
You get an API Key.
Paste this API key in the Text Translate API Key input field of Synbot Plugin configuration form and Save.
Syntax is
[x:Translate From="Source Language Code" To="Language to translate code"]Text to translate[/x:Translate]
[x:Translate From="Source Language Code"]Text to translate[/x:Translate]
[x:Translate To="Language to translate code"]Text to translate[/x:Translate]
[x:Translate]Text to translate[/x:Translate]
When the From parameter is omitted , source language for the text to translate will be automatically detected by the service.
When the To parameter is omitted , language to translate code will be automatically set with code of language used in the session - value of SIML User Variable Userlanguage
Language codes are those used and supported by the Microsoft TRANSLATE_TEXT API.
Tag will return the translated text value.
SIML User variable Languagedetected will be set with code of the source language detected.

SIML User variable FlagOKtranslate will be set to True or False depending on the translation was successful or not (In case not successful , some error messages
will be display in EZ Builder log console).
Exemple used in the Vision Description Query Model - File EZ Robot Request-Queries.siml
```
[Var Set="Visiondescription"][x:Translate][Var Get="Visiondescription" /][/x:Translate][/Var]
```

Twitter interface
[x:Twitter] tag
With [x:Twitter] tag , you can publish a tweet on the twitter bot account or send a twitter private message to a twitter user.
You need first to sign up for a specific twitter account for your robot and to configure the twitter stream input parameters in the Synbot Plugin Configuration Form.
(See advanced plugin configuration options before)
Syntax is
[x:Twitter Function="SEND" To = "Twitter user name"]Text to Send[/x:Twitter]
[x:Twitter Function="PUBLISH" File="Full Pathname for file attachment"]Text to Send[/x:Twitter]
If file parameter is omitted , no attachement will be published in the tweet message.
Tag will return True or False depending on the action was successful or not (In case not successful for example incorrect pathname for attachment, some error messages
will be display in EZ Builder log console).
File attachment is not today supported with the SEND function.
For Publish function , as it is asynchronous , a SIML Event will be raised when the operation is completed (succesfully or not) :
Event names are RETURNFROMTWEETPUBLISHINGOK if successful and RETURNFROMTWEETPUBLISHINGKO if not successful.
Models can be included in SIML files to define what to do when these events are raised.
For example:
```
[Var Set="Flagerr"][x:Twitter Function="PUBLISH" File=""][User Get="Textentered" /][/x:Twitter][/Var]
[Var Set="Flagerr"][x:Twitter Function="SEND" To=""][User Get="Textentered" /][/x:Twitter][/Var]
[If Var="Flagerr" Value="True"][User Set="bot_event_response" xml:space="preserve"]
[Random Get="" /]§[/User][/If]
[If Var="Flagerr" Value="False"][User Set="bot_event_response" xml:space="preserve"]
[User Get="NotOKmesssaved" /]§[/User][/If]
```

See detailed example in SIML File "EZ Robot request - Bot Response Message building"
Send mail interface
[x:Mail] tag
With [x:mail] tag , you can send a mail to an email adress , with or without an attachment file.
You need first create an email address and account for your robot and to configure the Bot email parameters in the Synbot Plugin Configuration Form.
(See advanced plugin configuration options before)
Syntax is
[x:Mail Function="SEND" To="email adress" File="Full Pathname for file attachment" Subject="subject text for the email"]Text to Send[/x:Mail]
If file parameter is omitted , no attachement will be attached to the message.
If subject parameter is omitted , message subject will be empty.

if function parameter is omitted , implicit SEND function will be used.
Tag will return True or False depending on the action was successful or not (In case not successful for example incorrect pathname for attachment, some error messages will be display in EZ Builder log console).
As SEND function is asynchronous , a SIML Event will be raised when the operation is completed (succesfully or not) :
Event names are RETURNFROMMAILSENDOK if successful and RETURNFROMMAILSENDOK if not successful.
Models can be included in SIML files to define what to do when these events are raised.
For example:

```
[Var Set="Flagerr"][x:Mail Function="SEND" To="" Subject="MAYA INFO" File=""][User Get="Textentered" /][/x:Mail][/Var]
```

See detailed example in SIML File "EZ Robot request - Bot Response Message building"

Find last file created or modified in a Windows folder

[x:Findfile] tag

With [x:Findfile] tag , you can retrieve name of the last created or modified file in a windows folder.
This tag can be used for example to retrieved file name for the last photo or video taken by your robot in order to send or publish it.
Syntax is

[x:Findfile Function="LAST" Filter="Filter for file to retrieve"]Path of folder to Search[/x:Mail]

Filter parameter value can be any file search filter - several filters can be set separated by |.
Example Filter="*.jpg*|.png|*.bmp" will retrieve only files with jpg, png and bmp extension.
If filter parameter is omitted , no filter will be apply.
Tag will return either name of file retrieved or empty value if no file found or if error occured.
For example:

```
[Var Set="filename"][x:Findfile Filter=""][Bot Get="EZmediafolder" /][/x:Findfile][/Var]
[If Var="filename" Value=""][Var Set="Flagerr"]1[/Var][/If]
```

See detailed example in SIML File "EZ Robot request - Bot Response Message building"

Syntactic analysis and control for a free text with structure [preposition] [article] [name]

[x:Decodeattribute]

With [x:Decodeattribute] tag , you can do a syntactic analysis and control for a free text part.
Syntactic form supported is [Prefix] [Preposition] [Article] [Name]
NOTE THAT THE TAG IS A BETA VERSION WHICH CAN HAVE MANY UPDATES IN NEXT VERSIONS
Several types of decoding and control can be specified depending on value of function parameter:
"F" - No decoding and no control done on the text except searching if text exist or in the plural form SIML Map name_pluriels_[language code]
If we detect that text is a plural form in this map - the singular entry will be returned and genre variable will be set to P as Plural
"0" or "1" - Text decoding is done - if "1" control is done with preposition and article authorization parameters
Search and suppressing of Prefix if Prefix parameter is set.

Search for preposition : as (comme) on (sur) to (à ,....) of (de, ...) about (au sujet, sur le sujet) during (pendant)
Search for article : a an the - un une le la les, ...
At the end of decoding steps - SIML variables are set :
. decodearticle with the article found (Empty if no article, A, THE, P(Plural article in french)
. decodepreposition with the preposition found (Empty if no preposition, TO, OF ,OF, AS, ABOUT)
. decodegenre with the genre found depending of article and existing or not of a plural form in SIML Map name_pluriels_[language code]
genre can be F(eminine) M(asculine) N(euter) I(not determined) or P(lural)
. decodename with the singular form of [name] after decoding
. decodenamewithde decodename form preceded by OF (de) preposition
. decodenamewitharticle decodename form preceded by the corresponding to genre article
Control of text decoding results is after done dépending on value of Article and Preposition Parameters.
. Preposition parameter is a word of 10 characters with value F(false) or T(rue) - T(rue) meaning authorized.
- Char 1 authorize or not text structure with no preposition.
- Char 2 authorize or not text structure with type TO preposition.
- Char 3 authorize or not text structure with type OF preposition.
- Char 4 authorize or not text structure with type AS preposition.
- Char 5 authorize or not text structure with type ON preposition.
- Char 6 authorize or not text structure with type ABOUT preposition.
- Char 7 authorize or not text structure with type DURING preposition.
- Char 8 to 10 are reserved for future use
. Article parameter is a word of 4 characters with value F(false) or T(rue) - T(rue) meaning authorized.
- Char 1 authorize or not text structure with no article.
- Char 2 authorize or not text structure with type AAN article.
- Char 3 authorize or not text structure with type THE article
- Char 4 authorize or not text structure with type PLURAL article.
Tag return True or False dépending the authorization and of the text decoding results.
If Preposition parameter is omitted , implicit value is all authorized (TTTTTTTTTT).
If article parameter is omitted , implicit value is all authorized (TTTT).
"2" - Same as 1 but with an additionnal control with the content of SIML map whose name is specified in the Map genreparameter.
Structure of this control map for each entry must be as
```
[MapItem Content="fr|lion" Value="%MM lion" /]
```

Entry Key is codelanguage|decodename and value content structured with char 1 % , char 2 genre of name , char 3 genre.
For example for a person name John, Entry will be "en|John" and value "%NM John" - as genre of name is N(euter) and person genre is M(asculine).
Tag will return F(alse) if key for codelanguage|decode name is not existing in the control map and when there are inconsistancies between genre decoded in text and genre values in the control map entries.
If Control is OK , value SIML variables decodegenre and decodegenreneutral are set with genre of name and genre found in the control map.
This kind of control is not critical in english language but is for french.
If you know for example that "lion" is a Masculine Genre Name , you cannot authorize text structure with "la lion" where presence of the "la" article induce a Feminine genre.
If function parameter is omitted , implicit value will be set to F.
Value of SIML Variable decodenameres is in any case set to initial text with some

transformations in order to be used to build bot responses
Examples
Text: "of the lion" will lead to decodepreposition = OF, decodearticle = THE decodename = lion, decodegenre = I
Text: "about John" will lead to decodepreposition = ABOUT, decodearticle = "" decodename = John, decodegenre = I
Tag syntax :
[x:Decodeattribute Type="Decoding type" Article="Article authorization word" Preposition="" Mapgenre="name of control SIML Map"]Text to decode[/x:Decodeattribute]
For example :
```
[Var Set="flagok"][x:Decodeattribute Type="" Article="" Preposition="" Mapgenre="genre_name"][User Get="freeattribute" /][/x:Decodeattribute][/Var]
```

Control Text operations
[x:Textcontrol]
With [x:Textcontrol] tag , you can do some control on a text string - Numeric , date format , ...
Syntax is
[x:Textcontrol Type="Control type mnemonic"]text string[/x:Textcontrol]
Return True if one of the control is successfull or False if all controls are not successful.
Control type is a text word giving which kind of controls to do.
If Character 1 if T(True) - control if text string is or not a numeric integer will be done.
Next characters are reserved for future use (Other controls)
Mnemonic values are specified in map typecontrol_freeattribute

```
[Map Name="typecontrol_freeattribute"]
[MapItem Content="NO" Value="FFFFFF" /]
[MapItem Content="NUMERIC" Value="TFFFFF" /]
[MapItem Content="FULLDATE" Value="FTFFFF" /]
[MapItem Content="NUMERICORFULLDATE" Value="TTFFFF" /]
[/Map]
```

```
[Var Set="Flagok"][x:Textcontrol Type="NUMERIC"][Var Get="namekey" /][/x:Textcontrol][/Var]
```

Syntactic Text operations
[x:Syntax]
With [x:Syntax] tag , you can do some Syntactic analysis or replacement on a text string
Syntax is
[x:Syntax Function="Action Type"]Input text string[/x:Syntax]
Tag Result is modified text string depending on Action Type.
Action Type replacepossessiveadjective do a syntactic replacement of possessive adjectives found in the input string - in english "My" is transformed in "Your" and "your" in "My" and the same type of replacement in french

```
[Var xml:space="preserve" Set="message_result"][x:Syntax Function="replacepossessiveadjective"][Var Get="message_result" /][/x:Syntax][/Var]
```

```

**Text operations**
**[x:Textop]**
With [x:Textop] tag , you can do some operations on a text string - Substring - replacing characters - splitting
Syntax is
**[x:Textop Function = "function"  Delimitor = "Delimitor" At = "position" Length = "length" Value = "Value" ]text string[/x:Textop]**
Return value depending on function :
Function = "Substring" implicit value - At and Length parameter Mandatory - Return Substring of input text at Position with Length Length
Function = "Split" - Delimitor parameter Mandatory - Character delimitor - Return Number of spliited element
Splitted text can be retrieved in SIML variables ressplit_n
Function = "Replace" - Parameter value is mandatory for replacing text  and At for position - Length is optionnal(Implicit length of replacing text)
Return Text modified with replacing text at position At on length characters

```
[Var Set="Nbsplit"][x:Textop Function="Split" Delimitor="|"][Match /][/x:Textop][/Var]
```
```

Step 11 : Pattern Reductions

Using Pattern reductions
In the Pattern Reductions SIML file , you can specify patterns which will be evaluate first for every User Input to the Bot.
Look at the example file delivered in which pattern reduction is used to specify several common syntax forms for a request.
COULD YOU TELL ME - CAN YOU GIVE US - I WOULD LIKE YOU TO GIVE - I JUST WANT YOU TELL ME - I WANT TO KNOW - COULD I KNOW - GIVE US - TELL ME - .....
User Input **I WANT TO KNOW WHAT IS YOUR NAME** will in this case switch to the evaluation of Pattern **WHAT IS YOUR NAME** due to the Goto tag
Variables values can be set in the Pattern Reductions Models in ordered to be tested in the switched Pattern Model.
Personnal framework example include automatic setting of some SIML variables Flags Typego1 to Typego15 , depending on the verb used.

```
[Map Name="Peux_Tu_en"]
[MapItem Content="can you" Value="T" /]
[MapItem Content="could you" Value="T" /]
[/Map]
[Map Name="Pourrai_je_en"]
[MapItem Content="can I" Value="T" /]
[MapItem Content="could I" Value="T" /]
[MapItem Content="can we" Value="T" /]
[MapItem Content="could we" Value="T" /]
[/Map]
[Map Name="Aimerai_voudrai_je_en"]
[MapItem Content="I would like" Value="T" /]
[MapItem Content="I wish" Value="T" /]
[MapItem Content="I just wish" Value="T" /]
[MapItem Content="I want" Value="T" /]
[MapItem Content="I just want" Value="T" /]
[/Map]
```

Different User input structures are taken into account in the example models :
. For kind of verbs configured in map CURRENT_VERB_1_EN
Phrase structure of type : can you give me , could you tell us , can you give , could you tell , I would like you give , I just want you tell , we wish you give
I would like you give me , I just want you tell us , we wish you give us , give me , tell us , give , tell
. For kind of verbs configured in map CURRENT_VERB_2_EN and Conjugate form im map CURRENT_VERB_2_FORM2_EN
Phrase structure of type : can I know , could we have , I would like to know , we wish to have , do you have , do you know
. For kind of verbs configured in map CURRENT_VERB_3_EN
Phrase structure of type : can you learn that , I would like you learn that, I just want you learn that , we wish you learn that , learn that.

```
[Map Name="current_verb_2_en"]
[MapItem Content="know" Value="%" /]
```

```
[MapItem Content="have" Value="%" /]
[/Map]
[Map Name="current_verb_2_form2_en"]
[MapItem Content="have you" Value="have" /]
[MapItem Content="do you have" Value="have" /]
[MapItem Content="do you know" Value="know" /]
[/Map]
```

Note than content value for conjugate form maps links to the key entry in the verb map.
Same kind of Maps (but suffixed with fr) are existing to configure french verbs :
current_verb_1_fr , current_verb_1_form1_fr , current_verb_1_form_fr , current_verb_2_fr , current_verb_2_form2_fr
current_verb_3_fr , current_verb_3_form1_fr , current_verb_3_form2_fr
In the pattern reductions example files you find also recognition in user input phrases of some greetings and politness formulas configured in map greetins and politness and combined with bot name.

```
[Map Name="greetings"]
[MapItem Content="Hi" Value="en" /]
[MapItem Content="Hello" Value="0" /]
[MapItem Content="Bonjour" Value="fr" /]
[/Map]
[Map Name="politeness"]
[MapItem Content="S'il te plait" Value="fr" /]
[MapItem Content="S'il vous plait" Value="fr" /]
[MapItem Content="Please" Value="en" /]
[/Map]
```

For example Input phrase : "Hi Maya could you give me your voltage please" would be redirected to the "XXXXXFROMPATTERNREDUCTION Your voltage" Pattern Model.
Note the XXXXXFROMPATTERNREDUCTION prefix indicating that a verb pattern reduction interpretation has bee done in the user input phrase.
Other Pattern reductions for EZ Commands verbs are included in EZ Robot request SIML File.

Using Pattern reductions - setting of some SIML variables Flags Typego1 to 15

Map **autho_numflag_verb** is used to configure flags TypeGoxx setting depending on the verb found in input pattern

```
[MapItem Content="1|dire" Value="%" /]
[MapItem Content="1|tell" Value="%" /]
[MapItem Content="1|say" Value="%" /]
[MapItem Content="1|savoir" Value="%" /]
[MapItem Content="1|connaître" Value="%" /]
[MapItem Content="1|connaitre" Value="%" /]
[MapItem Content="1|know" Value="%" /]
[MapItem Content="2|donner" Value="%" /]
[MapItem Content="2|connaître" Value="%" /]
[MapItem Content="2|connaitre" Value="%" /]
[MapItem Content="2|tell" Value="%" /]
[MapItem Content="2|give" Value="%" /]
```

Entry key is flag index|verb keyword , for example if we found the verb tell in input pattern , variable flag Typego1 will be set to 1.
Verbs are limited to verbs configured in pattern reduction example (maps CURRENT_VERB_1_EN , CURRENT_VERB_2_EN , CURRENT_VERB_3_EN and same suffixed by fr for french language)
Variable flag Typego is also Set to 1 if a verb pattern reduction is applied.
For example , For input "Give me your voltage" - Pattern activated will be "XXXXXFROMPATTERNREDUCTION1 your voltage" with Typego and Typego2 set to 1.
For Input "talk us your voltage" , pattern activated will be the same with Typego2 set to empty string.
In the "XXXXXFROMPATTERNREDUCTION1 your voltage" pattern model, we can so test typego2 variable and reject the incorrect input phrase "talk us your voltage"

```
[Model]
[Pattern]
[Item]XXXXXFROMPATTERNREDUCTION1 [MAP:DIRECT_QUESTION_OBJECT_EN] $[/Item]
[/Pattern]
[Response xml:space="preserve"]
[Think]
[If Var="Typego2" Value="1"]
[Var Set="temp1"][Map Get="direct_question_object_en"][Match /][/Map][/Var]
[Var Set="keyobject"][Text WordAt="2"][Var Get="temp1" /][/Text][/Var]
[Var Set="Compattribute"][Match At="2" /][/Var]
[Goto xml:space="preserve"]XXXXXQUERYEZ [Var Get="keyobject" /][/Goto]
[/If]
[ElseIf Var="Typego11" Value="1"][Goto
xml:space="preserve"]XXXXXFROMPATTERNREDUCTION2 [Match /] [Match At="2" /][/
Goto][/ElseIf]
[/Think]
[/Response]
[/Model]
[Map Name="direct_question_object_en"]
[MapItem Content="your current voltage" Value="%M VOLTAGE" /]
[MapItem Content="your voltage" Value="%M VOLTAGE" /]
[MapItem Content="your temperature" Value="%F PROCESSORTEMPERATURE" /]
[MapItem Content="your processor temperature" Value="%F PROCESSORTEMPERATURE" /]
[/Map]
```

Example can be found in EZ Robot Request - Queries SIML File.
Note The test included for Typego11 - in order to redirect to correct pattern because entries in the 2 maps VARIABLE_VALUE_EN and direct_question_object_en can be the same.
and for this entries pattern [Item]XXXXXFROMPATTERNREDUCTION1 [MAP:VARIABLE_VALUE_EN] * TO *[/Item] will never be activated.

```
[Model]
[Pattern]
[Item]XXXXXFROMPATTERNREDUCTION1 [MAP:VARIABLE_VALUE_EN] * TO *[/Item]
[Item]XXXXXFROMPATTERNREDUCTION1 [MAP:VARIABLE_VALUE_EN] * ON *[/Item]
[Item]XXXXXFROMPATTERNREDUCTION2 [MAP:VARIABLE_VALUE_EN] * TO *[/Item]
[Item]XXXXXFROMPATTERNREDUCTION2 [MAP:VARIABLE_VALUE_EN] * ON *[/Item]
[/Pattern]
[Response xml:space="preserve"]
```

```
[Think]
[If Var="Typego11" Value="1"]
[Var Set="Temp"][Match At="2" /][/Var]
[Var Set="Temp2"][x:EZvar Get="" /][/Var]
[x:EZvar Set="" Value="" /]
[User Think:Set="bot_event_response"]Value of variable [Var Get="Temp" /] has been set
to [Match At="3" /][/User]
[/If]
[/Think]
[/Response]
[/Model]
```

Step 12 : SIML Tips and Tricks

Using project Editor Syn Bot Studio
Be CAREFUL :
in editing - there 's no Ctrl-Z feature - so you cannot go back and cancel your last changes.
There's no SAVE AS feature for your project
So do regularly some backups of your SIML Project Folder in order to be able to backup this project to an older situation.
Build Project (Project - Build Project or F5) make a syntax analysis of your project before building and saving your SIML Project.
It works in 2 or 3 passes in order to detect first level of syntax errors.
When you got an error , error message points to line and position of error but not to the SIML file in which is located the Error.
If you have made changes to several SIML files of your project , it can be hard to identify in which file is located the syntax error.
So I recommend you do as often as possible 2 or 3 times F5 to be sure to track this firtst level of errors when you are making changes to your SIML Project.
A deeper syntax analysis is done with Project Analyse Project - you need to correct only the detected High Severity errors.
As SIML is XML , it is very sensitive.
Some syntax errors are not today completely tracked in Build and Analyse Project Features.
You need to be careful in the SIML tags attributes Syntax.

```
 [Response]

[If User="flagtest" Value="testOK"]It's working fine all is OK [/If]
[/Response]
```

```
[Response]

[If User="flagtest" value="testOK"]It's working fine all is OK [/If]
[/Response]
```

In this example , you will not get any syntax error, but at runtime the second form of code will never work correctly cause the If block will never be executed.
Only because the attribute **value** is not correct and must be **Value**
And it can be tricky to debug this kind of error.
Tools - SIML Playground give you access to an integrated SIML Language Tutorial.
SIML : Using Filters
Filters can be set in Normalizations SIML File.
When you use filters , You replace all Words defined inside the filter tag by the content specified in the parameter value of the filter tag.
Replacement is done before the bot interpretation.

```
[Filter Value="camera"]
[Word]cam[/Word]
[/Filter][Code]
```

In this case, word "cam" in a input user phrase will be replaced by "camera".
But be CAREFUL in the Models - if you define a pattern
[Code][Model]
[Pattern]
[Item]TURN YOUR CAM[/Item]
[/Pattern]
[Response xml:space="preserve"]
I turn my cam
[/Response]
[/Model]
```
It will never been activated - User input "Turn your Cam" will be transformed with the filter in "Turn Your Camera"  before Bot interpretation ....
Filters can be used to suppress ambigous interpretation by the bot -
In the example - you can have "turn on the right" on a syntaxic form [Verb] [Movement Direction] with Verb = TURN and Movement Direction = ON THE RIGHT
You can also have "turn on your camera" on a syntaxic form [Verb] [Complement] with Verb = TURN ON and Complement = YOUR CAM
If we do nothing verb will be priorized and the first phrase input will never be recognized.
To work fine we include the following filters
```
 [Filter Value="Turn to the right"]
[Word]turn on the right[/Word]
[Word]turn on right[/Word]
[/Filter]
[Filter Value="Turn to the left"]
[Word]turn on the left[/Word]
[Word]turn on left[/Word]
[/Filter]
[Filter Value="Turn to port"]
[Word]turn on port side[/Word]
[/Filter]
[Filter Value="Turn to starboard"]
[Word]turn on starboard[/Word]
[/Filter]
```

**SIML : Using Set and Map tags**
**In [Pattern] for a collection of words or sentences use  the Map tag rather than the Set Tag**
Example with Set
```
[Set Name="Color"]
[Item]Red[/Item]
[Item]Green[/Item]
[Item]Blue[/Item]
[/Set]
```

```
Model]
[Pattern]I LIKE THE COLOR [COLOR][/Pattern]
[Response]Aha! So you love [Match /].[/Response]
[/Model]
```
```

```

The sentences I LIKE THE COLOR RED , I LIKE THE COLOR GREEN , I LIKE THE COLOR BLUE will match the pattern.
But the pattern below lead to a syntax Error in Analyse project : the word RED is already adressed by the SET COLOR.

```
[Model]
[Pattern]DO YOU KNOW WHAT IS THE RED MENACE[/Pattern]
[Response]Yes Guy , the communist menace[/Response]
[/Model]
```
In a large project it can become very tricky ....
So prefer using Map tag

```
[Map Name="Color"]
[MapItem Content="Red" Value="T" /]
[MapItem Content="Green" Value="T" /]
[MapItem Content="Blue" Value="T" /]
[/Map]
```

```
[Model]
[Pattern]I LIKE THE COLOR [Map:COLOR][/Pattern]
[Response]Aha! So you love [Match /].[/Response]
[/Model]
```

And all will be working fine
SIML : Using the Map tag
As in Set , you cannot have in a Map duplicate for Map entries contents.
Be CAREFUL if you use the Reverse attribute for a Map.
In this case , There cannot have also duplicates for content values of Map entries.
In case of duplicates , value corrresponding to first entry will be return with Get.
Be CAREFUL to the order of entries in the Map - Example below will lead to unexpected results

```
[Map Name="Color"]
[MapItem Content="Red" Value="1" /]
[MapItem Content="Red Blue" Value="2" /]
[MapItem Content="Red Brown" Value="3" /]

[MapItem Content="Green" Value="4" /]
[MapItem Content="Blue" Value="5" /]
[MapItem Content="Red" Value="6" /]

[/Map]/
```

```

[Model]
[Pattern]I LIKE THE COLOR [Map:COLOR][/Pattern]
[Response]Aha! So you love [Match /] and index in color table is [Map Get="COLOR"][Match /][/Map][/Response]
[/Model]
```

I like the color Red Blue : Aha! So you love red blue and index in color table is 1 5
I like the color Red Brown : Aha! So you love red blue and index in color table is 1 Brown
If you change the order of entry in the map

```
[Map Name="Color"]
[MapItem Content="Red Blue" Value="2" /]
[MapItem Content="Red Brown" Value="3" /]

[MapItem Content="Red" Value="1" /]
[MapItem Content="Green" Value="4" /]
[MapItem Content="Blue" Value="5" /]
[MapItem Content="Red" Value="6" /]

[/Map]
```

Results will be as expected :
I like the color Red Blue : Aha! So you love red blue and index in color table is 2
I like the color Red Brown : Aha! So you love red blue and index in color table is 3
Aha! So you love red blue and index in color table is 2

SIML : Using the Map tag - Get the value of a Map

[Map Get="name of Map"]Entry Content to search[/Map] returns the value corresponding to the Entry content to search.
If the Entry is not found in the Map , result will be the Entry Content specified.
[Map Get="COLOR"]Yellow[/Map] will return Yellow

SIML : Using the Bind tag

Binding allows values of adapters to be mapped to unique keys that upon evaluation get replaced with the returned value of the adapter. Binding is very useful to assign values to Tag attributes
Example Comparing 2 Bot variables values

```
[If Var="Controlrobottype" Value=""][User Set="indrobot"][/User][/If]
```

To compare User variables value Controlrobottype and Controlrobottype1

SIML : Using the Goto tag

Once you have caught a pattern at some point you may want to redirect the pattern-search to a different value. The value of the Goto elementt tells the Bot that it should recursively use the pattern defined within it and search for a response.
Goto element plays a major role in breaking down complex sentences into simpler ones.
The Goto tag is very powerful but remember always that after exécution of the called model, control is set back to the caller
So for example the last reponse will not be set to the goto pattern reponse but to the caller response.

SIML : Using If Else and Switch Tags

Be careful as nested If or nested Switch can have impredictible results.
You can better nest alternate If and Switch tags
```

Step 13 : EZ Robot Commands SIML Framework

In "EZ Robot Request.SIML" file (one of the file of the example SIML project) is included a Framework to make easier interpretation of input text to command EZ Builder actions.
French and English Syntax are now supported.
But Framework structure enable easily to add other languages syntax interpretation.
Framework configuration is done with siml Map objects.
Bot Response messages to User Input text are formatted in "EZ Robot request - Bot Response Message building" SIML file
The following Input Text syntax can be analysed and interpreted to generate a Bot Response with or without EZ Robots Commands.
You need first a **[Verb]**.
If no [Verb] in the input text , you can configure verb to use implicitely in Maps **synonym_global_fr** or **synonym_global_en** - only with [Movement Direction] or [Position]
for example you can configure a synonym for input text FORWARD as MOVE FORWARD , ...
Global Robot Movement Control
[Verb][Movement Direction]
Example : "Move Forward" , "Turn Left" , ...
[Verb][Movement Direction] [Secondary Movement Direction]
Example : "Move Backward to the left" " Turn Left to the rear", ...
You can include for global Robot Movement **[Speed Adjective]** and **[Movement Duration]**
Example : "Go Forward at full speed" , "Move back during 670 milliseconds" , turn right forward slowly" , 'turn fast left backward" , ...
[Action on speed] or **[Verb] [Action on speed]** in order to command the Robot movement speed
For Example : "Slow down" , "Go quicker" , ...
Some [Verb] can be configured with a direct synonym to a [Movement Direction] or an [Action on speed]
For example the verb "GO BACK" can be directly linked to the Movement Direction "BACK" the verb "ACCELERATE" can be directly linked to the action on speed "FASTER"
Robot Parts Movements and Actions Control
[Verb] [Complement] where complement can be a robot Part HEAD , ARMS , LEFT ARM , ...or a Robot Feature or Behaviour PHOTO , VIDEO , RECOGNITION , COLOR RECOGNITION , VIDEO RECORDING , MAIL , TWITTER MESSAGE , ...
For exemple : "center your head" , "open left hand" , "pause the Camera", "Take a photo" , "Start Color recognition" , "Send twitter message" , "Start video recording" , ...
You can add a **[Movement Duration]** - for example "Take a video during 10 seconds"
[Verb] [Complement] [Movement Direction]
For exemple : "lower your arms" , "point your camera to the right" , ...
[Verb] [Complement] [Movement Direction] [SecondaryMovement Direction]
For exemple : "move the head downward to the right" , ...
[Verb] [Complement] [Attribute]
[Verb] [Attribute] [Complement]
For example : "Balance the head from right to left" , "swing your left arm from bottom to top"
[Verb] [Complement] [Free Attribute]
[Verb] [Free Attribute] [Complement]
[Verb] [Complement] [Attribute] [Free Attribute]
[Verb] [Free Attribute] [Complement] [Attribute]
[Verb] [Free Attribute] [Attribute] [Complement]
[Verb] [Attribute] [Complement] [Free Attribute]

For example : "send a mail to John" - John is a free attribute and will be decoded depending on a map configured to store the email addresses.
Some [Verb] can be configured with a direct synonym .
For example the verb "LOWER" is directly linked to the Movement Direction "DOWN".
Synonyms can be a Combination of [Movement Direction][Complement] [Position] and [Position] or anything you want if model for the pattern is defined in SIML files.

Robot Position Action

Position is a position or a sequence of positions of the Robot for example the Calibrate position , the sitting position , the push-ups sequence

[Verb] [Position]

For example : "Put yourself in the sitting position" , "do some push-ups" , ...
If no [Verb] in the input text , you can configure verb to use implicitely in Maps **synonym_global_fr** or **synonym_global_en**
For example IN INITIAL POSITION is equivalent to PUT YOU IN INITIAL POSITION

[Verb] [Complement] [Position]

For example : "Switch the cam Off" , "Put your arms on the cross"
Some [Verb] can be configured with a direct synonym to a [Position]
For example the verb "SIT" is directly linked to the position "SITTING"

[Verb] [Complement] [attribute] [Position]

[Verb] [Position] [Free Attribute]

Other actions

[Verb] without any complement
For example : "Smile"

[Verb] [Free Attribute]

For example : "Sing Happy birthday" , "Dance on Happy Birthday"

(Verb] structure

Below is the SIML Patterns used in English for all [Verb] structure ,and the associated Sets.
Only one verb form configured in the map MOVEMENT_VERB_1_EN is necessary.
The Bot will recognize in the same manner for the verb "Move" : "Move" , "Can you move", "Could you move" , "I would like you move" , "I would like you to move" , "We just want you move" , ...
Map content can be extended with new entries.
Structure is more complex in french , as we need for one verb 3 forms :
Example for the "move" in french "aller"
"Move" : " Va"
"Can you move" : "Peux tu aller"
"I would like you to move" : "J'aimerai que tu ailles"
Syntax can be configured in SIML file EZ Robot request and in the SIML Maps :

```
[Map Name="Peux_Tu_en">
[MapItem Content="can you" Value="T" />
[MapItem Content="could you" Value="T" />
[/Map>
[Map Name="Aimerai_voudrai_je_en">
[MapItem Content="I would like" Value="T" />
[MapItem Content="I wish" Value="T" />
[MapItem Content="I just wish" Value="T" />
[MapItem Content="I want" Value="T" />
[MapItem Content="I just want" Value="T" />
[/Map>
[Map Name="Aimerai_voudrai_nous_en">
[MapItem Content="we would like" Value="T" />
[MapItem Content="we wish" Value="T" />
[MapItem Content="we just wish" Value="T" />
```

```
[MapItem Content="we want" Value="T" />
[MapItem Content="we just want" Value="T" />
[/Map>
```

Robot type and Robot Type Authorization
Map **typerobot** is used to configure the robot types recognized
```
[Map Name="typerobot"]
[MapItem Content="JD" Value="1" /]
[MapItem Content="SIX" Value="3" /]
[MapItem Content="ROLLI" Value="2" /]
[MapItem Content="ADVENTUREL" Value="4" /]
[/Map]
```
Content must link with the Robot Type configured in the SynBot Plugin
Value is the corresponding robot index - a maximum of 10 different robot types can be configured.
Map **Robot_autho_model** is used to configure robot types authorization models
```
[Map Name="Robot_autho_model"]
[MapItem Content="ALL" Value="TTTTTTTTTT" /]
[MapItem Content="NO" Value="FFFFFFFFFF" /]
[MapItem Content="WITHCAM" Value="TTTTTTTTTT" /]
[MapItem Content="WITHHEAD" Value="TFFFFFFFFF" /]
[MapItem Content="WITH2LEGS" Value="TFFFFFFFFF" /]
[MapItem Content="WITH6LEGS" Value="FTFFFFFFFF" /]
[MapItem Content="WITHLEGS" Value="TTFFFFFFFF" /]
[MapItem Content="WITHWHEELS" Value="FFTTFFFFFF" /]
[MapItem Content="WITH2ARMS" Value="TFTFFFFFFF" /]
[MapItem Content="WITH2HANDS" Value="TFFFFFFFFF" /]
[MapItem Content="WITH2GRIPPERS" Value="TFTFFFFFFF" /]
[MapItem Content="WITHFIXEDCAM" Value="FFFTFFFFFF" /]
[MapItem Content="WITHMOVINGCAM" Value="TTTFTTTTTT" /]
[MapItem Content="JD" Value="TFFFFFFFFF" /]
[/Map]
```
Content is an authorization model mnemonic which will be used in other maps to specify type of robot authorization.
Value is a ten characters string and for each character an authorization F False or T True. Position of the character in the string links to robot index (for example first character for robot index 1 JD)
Movement Direction
Maps **movement_direction_fr** and **movement_direction_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Movement direction)

```
[MapItem Content="to the right" Value="% RIGHT" /]
[MapItem Content="on the right side" Value="% RIGHT" /]
```

On a main entry content (Char 1 = %)
Second block is a mnemonic link to the Map **movement_direction_global** in which are defined for the mnemonic link entry global configuration parameters (valid for every language used).

```
[Map Name="movement_direction_global"]
[MapItem Content="LEFT" Value="% L" /]
[MapItem Content="RIGHT" Value="% R" /]
[MapItem Content="FORWARD" Value="% F" /]
[MapItem Content="BACKWARD" Value="% B" /]
[MapItem Content="UP" Value="% U" /]
[MapItem Content="DOWN" Value="% D" /]
[/Map]
```

This map doesn't need to be changed.
The second character is just a character mnemonic for the movement direction F Forward B Backward L Left R Right U Up D Down
Direct calls without a preceding [verb] must be configured in the SIML maps **synonym_global_fr** or **synonym_global_en** depending on the language used.
Each entry authorized in direct call must refererenced in these maps - value for each entry give the text synonym to use with this direct call (including [verb].

```
[MapItem Content="left" Value="move left" /]
```

Configuring Verb , Complement , Position and Attribute
SIML Maps are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input in order to generate a Bot Response to command Robots.
Building Bot response is done in the SIML File : **EZ Robot request - Bot Response Message building**.
You will need to insert SIML codes in this file depending on the structure and the content of user Input Phrase.
Each entity 'Verb , Complement , Position and Attribute" if recognized in the input phrase will have in configuration maps a Bot Building ID or BBID which can be retrieved in SIML file to build the Bot Response. - ***See Next for more détails.***
Authorized combinations between [Verb][Complement][Attribute] and [Position] in the Input Phrase will be configured in a specific Map **combination_verb_comp_att_pos**
Each entity 'Verb , Complement , Position and Attribute" if recognized in the input phrase will have in configuration maps an AuthoID which will be used in key of this combination Maps.
This structure allows to regroup entities values with the same level of authorization.
To make easier the configuration :
BBID and AuthoID are the same for [Attribute]
They can be different for [Verb] [Complement] and [Position].

Managing Mandatories Argument in user input phrase
We can specify in [verb] or [Complement] Configuration Maps if additionnal [Movement Direction] , [Complement] [Attribute] [Position] are required (Mandatory) or not.
We can also specify in the Authorized Combination Map if [Movement Direction] , [Attribute] , [Free Attribute] and or [Duration] are possible(Authorized) or required (Mandatory) for the Combination.
When parameter value is Mandatory and is not found in the User Input Phrase , SIML Command Framework include standard dialogs with User in order to get the missing value.
Models are located in SIML File EZ Robot Request - Ask precision dialogs.
Example with "lift your left arm"
Lift need a complement so a first dialog is occuring to get this complement.
lift your arm is ambigous so a second dialog is occuring to get the non ambigous complement value

Standard dialogs message are coded in SIML File
EZ Robot Request - Ask precision dialogs
where you can modify them.
```
      [Switch Var="Typeask"]
[Case Value="1"][User xml:space="preserve" Set="bot_event_response"]can you clarify in what direction I should [User Get="usedverb" /] [User Get="verbcomplement" /][/User][/Case]
[Case Value="2"][User xml:space="preserve" Set="bot_event_response"]You ask me to [User Get="usedverb" /] [User Get="verbcomplement" /] can you clarify which one[/User][/Case]
[Case Value="4"][User xml:space="preserve" Set="bot_event_response"]What do you want I [User Get="usedverb" /][/User][/Case]
[Case Value="5"][User xml:space="preserve" Set="bot_event_response"]How do you want I [User Get="usedverb" /][/User][/Case]
[Case Value="6"][User xml:space="preserve" Set="bot_event_response"]You ask me to [User Get="usedverb" /] [User Get="verbcomplement" /] can you clarify how[/User][/Case]
[Case Value="7"][User xml:space="preserve" Set="bot_event_response"]You ask me to [User Get="usedverb" /] [User Get="verbcomplement" /] but how many time[/User][/Case]

```
several kinds of situation are included in the Framework (Value of the AskPrecision Variable but don't care) :
1 - Request for [Movement Direction]
2 - Ambigous value for [Complement] - Request for additionnal information
4 - Request for [Complement]
5 - Request for [Position]
6 - Request for [Attribute]
7 - Request for [Duration]
8 - Publishing Message - will request input by the user of a free text message which will be the message content to publish
9 - Sending Message - will request input by the user of a free text message which will be the message content to send
11 - Request for [Free attribute] example request song name with the Sing Verb
20 - Request for asking if user want or not more answers (if avalaible) after a Wikipedia Search
User answer can be Yes or No.
This model will be extend in next versions to other situations where User Yes or No Answer is awaited.
SIML file **Textinput** contains models used to be able to manage Free text user input (Case 8 , 9 and 11)
SIML Maps **Action_side_fr** and **Action_side_en** are used to configure the authorize user answers in case 2 (Left or right)
SIML Maps **Yesorno_fr** and **Yesorno_en** are used to configure the authorize user answers in case 20.

Configuring Movement Speed]
For Global Robot Movement direction control commands ,
a [Speed Adjective] can be used to define movement speed.
Maps **move_speed_fr** and **move_speed_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Speed Adjective)

```
[Map Name="move_speed_en"]
[MapItem Content="SPEEDNORMAL" Value="NORMAL" /]
[MapItem Content="normally" Value="NORMAL" /]
[MapItem Content="very slowly" Value="VERYSLOW" /]
[MapItem Content="slowly" Value="SLOW" /]
[MapItem Content="at full speed" Value="FULL" /]
[MapItem Content="full speed" Value="FULL" /]
[MapItem Content="quickly" Value="FAST" /]
[MapItem Content="fast" Value="FAST" /]
[MapItem Content="very fast" Value="VERYFAST" /]
[MapItem Content="very quickly" Value="VERYFAST" /]
[/Map]
```

Value is a one word mnemonic link to the Map **move_speed_global** in which are defined for the mnemonic link entry speed parameters (valid for every language used).

```
 [Map Name="move_speed_global"]
[MapItem Content="NORMAL" Value="100" /]
[MapItem Content="LOWNORMAL" Value="80" /]
[MapItem Content="SLOW" Value="60" /]
[MapItem Content="VERYSLOW" Value="20" /]
[MapItem Content="FULL" Value="255" /]
[MapItem Content="FAST" Value="160" /]
[MapItem Content="VERYFAST" Value="200" /]
[MapItem Content="VERYVERYFAST" Value="230" /]
[MapItem Content="ALMOSTNULL" Value="2" /]
[/Map]
```

For each Entry is specify the speed value corresponding between 1 to 255 (See EZ Builder Documentation)
Movement Speed Command]
Maps **move_speed_fr** and **move_speed_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Action on speed]

```
[Map Name="action_speed_en"]
[MapItem Content="much slower" Value="D 50" /]
[MapItem Content="much faster" Value="I 50" /]
[MapItem Content="much quicker" Value="I 50" /]
[MapItem Content="faster" Value="I 20" /]
```

```
[MapItem Content="quicker" Value="I 20" /]
[MapItem Content="slower" Value="D 20" /]
[MapItem Content="slow down" Value="D 20" /]
[MapItem Content="more gently" Value="D 20" /]
[MapItem Content="more slowly" Value="D 20" /]
[MapItem Content="full speed" Value="X 255" /]
[/Map]
```

Word 1 is a character indicating if action is Decreasing D or Increasing I speed - Value X is reserved for future use.
Word 2 is the speed increment or decrement
when several successive commands are input by user , control is done to keep speed values between 0 and 255.

Configuring Movement Duration]

Structure of [movement duration] is [During adverb] [number] [timing unit] for example "during 2200 milliseconds"
[During adverb] is configured for french and english language in SIMLMaps pendant_durant_fr and pendant_durant_en

```
[Map Name="pendant_durant_en"]
[MapItem Content="For" Value="T" /]
[MapItem Content="During" Value="T" /]
[/Map]
```

[number] is defined as a regex in regex SIML file @NOMBRE1_999999
[timing unit] is configured for french and english language in maps timing_fr and timing_en

```
  [Map Name="timing_en"]
[MapItem Content="milliseconds" Value="1" /]
[MapItem Content="milli seconds" Value="1" /]
[MapItem Content="seconds" Value="1000" /]
[MapItem Content="second" Value="1000" /]
[MapItem Content="minutes" Value="60000" /]
[MapItem Content="minute" Value="60000" /]
[/Map]
```

Content entry give the milliseconds value for the timing unit.

Verb
Maps **movement_verb_1_fr** and **movement_verb_1_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Verb)

```
   [MapItem Content="go" Value="%F NO GO" /]
[MapItem Content="move" Value="%F NO GO" /]
[MapItem Content="roll" Value="%F NO ROLL" /]
[MapItem Content="walk" Value="%F NO WALK" /]
[MapItem Content="run" Value="%T NO WALK FAST" /]
[MapItem Content="turn off" Value="switch off" /]
[MapItem Content="turn on" Value="switch on" /]
[MapItem Content="turn" Value="%F NO TURN2" /]
[MapItem Content="lift" Value="%F DIR LIFT" /]
[MapItem Content="raise" Value="lift" /]
[MapItem Content="switch off" Value="%F POS STOP" /]
[MapItem Content="switch on" Value="%F POS START1" /]
```
 If the first character of Value is not % , entry will be linked to the value.
For example "raise" will be equivalent to "lift" which is the main entry.
On a main entry content
Word 1
Character 1 = %
Character 2 specify if or not a parameter will be present in Word 4 - T True if present , F False if not.
Word 4 if present is a parameter value - this value can be retrieved when building Bot Response Message by value of **User Variable Verbargument**
This argument if present for Movement Direction Verb can be interpreted as movement speed mnemonic - Map move_speed_global - example is verb Run linking to the FAST mnemonic
Word2 is used to specify if [Verb] will be interpreted with a Synonym Command - Its a mnemonic configured in Synonym_autho_model Map (See after).
If no synonym mnemonic is NO
Word 3 is a mnemonic link to the Map **movement_verb_group** in which are defined for the mnemonic link entry global configuration parameters (valid for every language used).

```
[MapItem Content="GO" Value="% DIR FBLR FBLR SPEED1 ALL GO MOVE" /]
[MapItem Content="GO1" Value="% DIR FBLR FBLR SPEED1 ALL GO1 MOVE" /]
[MapItem Content="ROLL" Value="% DIR FBLR FBLR SPEED1 WITHWHEELS MOVE MOVE" /]
[MapItem Content="WALK" Value="% DIR FBLR FBLR SPEED1 WITHLEGS WALK MOVE" /]
[MapItem Content="TURN1" Value="% DIR LR FB SPEED2 ALL MOVE MOVE" /]
[MapItem Content="GOFORWARD" Value="% DIR F LR SPEED1 ALL GO1 MOVE" /]
[MapItem Content="GOBACK" Value="% DIR B LR SPEED1 ALL GO1 MOVE" /]
[MapItem Content="TURN2" Value="% DIR_ACOMP_APOS LR FB SPEED2 ALL TURN TURN" /]
[MapItem Content="MOVE6" Value="% DIR_ACOMP FBLR FBLR SPEED1 ALL GO1 MOVE" /]
[MapItem Content="MOVE7" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVE MOVE" /]
```

[MapItem Content="LIFT" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVE MOVE" /]
[MapItem Content="LOWER" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVE MOVE" /]

```

Value of Content entry is the mnemonic for the verb.
On each entry content
Word 1 : %
Word 2 is used to specify some informations about the verb command structure with a mnemonic linked to **verb_autho_model** map.

```

[MapItem Content="NO" Value="FFFF" /]
[MapItem Content="DIR" Value="FFFT" /]
[MapItem Content="MCOMP" Value="FMFF" /]
[MapItem Content="ACOMP" Value="FAFF" /]
[MapItem Content="DIR_ACOMP" Value="FAFT" /]
[MapItem Content="DIR_ACOMP_APOS" Value="FAAT" /]
[MapItem Content="MPOS" Value="FFMF" /]
[MapItem Content="APOS" Value="FFAF" /]
[MapItem Content="MCOMP_APOS" Value="FMAF" /]
[MapItem Content="MCOMP_MPOS" Value="FMMF" /]
[MapItem Content="ACOMP_APOS" Value="FAAF" /]
[MapItem Content="DIRECT" Value="TFFF" /]

```

Char 1 of Value is F False or True if we want to specify a [Verb] or [Verb][free attribute] Command structure for example SING a song.
Char 2 specify if [Complement] is M Mandatory A Authorized or F Forbidden in the command.
Char 3 specify if [Position] is M Mandatory A Authorized or F Forbidden in the command.
Char 4 specify if [Movement Direction] is Authorized T or not F in the command.
For example if you want that the verb authorize only a [verb][complement]command structure you choose as mnemonic **MCOMP** - [Complement] will be Mandatory and [Movement direction] [Attribute] and [Position] are not Authorized for Example CENTER mnemonic.
Word 3 is used to specify which Movement Directions are Authorized for [Verb] - It can be overwrited at the Complement Level and at the global Combination Level.
Word 4 is used to specify which Secondary Movement Directions are Authorized for [Verb] - It can also be overwrited at the Complement Level and at the global Combination Level.
Word 3 and Word 4 are mnemonic for movement directions authorizations Linked to **movement_autho_model** map.

```
[Map Name="movement_autho_model"]
[MapItem Content="ALL" Value="TTTTTT" /]
[MapItem Content="NO" Value="FFFFFF" /]
[MapItem Content="EMPTY" Value="FFFFFF" /]
[MapItem Content="FLRUD" Value="TFTTTT" /]
[MapItem Content="FBLUD" Value="TTTFTT" /]
[MapItem Content="FBRUD" Value="TTFTTT" /]
[MapItem Content="FBLRD" Value="TTTTFT" /]
[MapItem Content="FBLR" Value="TTTTFF" /]
```
```

[MapItem Content="LRUD" Value="FFTTTT" /]
[MapItem Content="FLRD" Value="TFTTFT" /]
[MapItem Content="FBUD" Value="TTFFTT" /]
[MapItem Content="FLR" Value="TFTTFF" /]
[MapItem Content="LR" Value="FFTTFF" /]
[MapItem Content="FB" Value="TTFFFF" /]
[MapItem Content="UD" Value="FFFFTT" /]
[MapItem Content="L" Value="FFTFFF" /]
[MapItem Content="R" Value="FFFTFF" /]
[MapItem Content="D" Value="FFFFFT" /]
[MapItem Content="U" Value="FFFFTF" /]
[MapItem Content="F" Value="TFFFFF" /]
[MapItem Content="B" Value="FTFFFF" /]
[/Map]
```

Value is a 6 characters Word - Each character  define the authorization or not for a movement direction T Authorized F  Not Authorized - Char 1 Forward - Char 2 Backward - Char 3 Left - Char 4 Right - Char 5 Up - Char 6 Down
For example mnemonic **R**  authorize only Movement Direction to the Right.
Word 5 is only used with [Verb] [Movement Direction] command structure (Otherwise EMPTY) - It's a mnemonic to **verb_direction_autho_model** Map.
```
[Map Name="verb_direction_autho_model"]
[MapItem Content="NO" Value="FF" /]
[MapItem Content="SPEED1" Value="T1" /]
[MapItem Content="SPEED2" Value="T2" /]
[/Map]
```

Char 1 specify if  [Movement Duration] is authorized T or not F.
Char 2 specify if [Speed Adjective]  are authorized 1 or 2 or not F
1 or 2 specify the way [Action Speed] will act - 1 same action on left and right speed - 2 - action on the difference between left and right speed.
Word 6 specify the Robot Type authorization for the verb - mnemonic linked to Robot Type authorization Map.
Word 7 is the Verb AuthoID
Word 8 is the Verb BBID
For example
```
[MapItem Content="TURN2" Value="% DIR_ACOMP_APOS LR FB SPEED2 ALL TURN TURN" /]
```

[Complement]  , [Position] and [Movement Direction] are Authorized and if [Movement Direction] is present only Left and Right Direction are Authorized for main Direction and Forward and Backward for the Secondary Direction.
The Verb BBID and AuthoID is TURN and at the verb level All Robot types are authorized.
Turn head to the Left - Turn left forward - Turn  cam on - will be recognized
In English language two complementary map are used for verb output form exceptions
**movement_verb_1_form3_en**  when the "I verb" is not correct - example with "Put you" verb "I put you" is incorrect and must be "**I'm putting me[b]"**
**[b]movement_verb_1_special_en** - for the verb "send me" the form "send you" when the command is not successful "I cannot send you".
**French Verb Configuring**
```

For french language , several additionnal Maps need to be configure :
for verb input forms **movement_verb_1_form1_fr** , **movement_verb_1_form2_fr** - example
verb "Aller " "Peux tu **aller** " "J'aimerai que tu **ailles** " and "**va** "
for verb output forms **movement_verb_1_form3_fr**and just with exceptions
movement_verb_1_form4_fr[/b} and [b] movement_verb_1_special_fr[/b} - example
verb "Aller " "[b]je vais " "dans quelle direction veux tu que**j'ailles** "
Verb Configuring synonym

```
[MapItem Content="switch off" Value="% POS STOP1" /]
```

Word2 of main entry in the Maps **movement_verb_1_fr** and **movement_verb_1_en** is used to specify if [Verb] will be interpreted with a Synonym Command - Its a mnemonic configured in Synonym_autho_model Map.

```
 [Map Name="verb_autho_model"]
[MapItem Content="NO" Value="FFFF" /]
[MapItem Content="DIR" Value="FFFT" /]
[MapItem Content="MCOMP" Value="FMFF" /]
[MapItem Content="ACOMP" Value="FAFF" /]
[MapItem Content="DIR_ACOMP" Value="FAFT" /]
[MapItem Content="DIR_ACOMP_APOS" Value="FAAT" /]
[MapItem Content="MPOS" Value="FFMF" /]
[MapItem Content="APOS" Value="FFAF" /]
[MapItem Content="MCOMP_APOS" Value="FMAF" /]
[MapItem Content="MCOMP_MPOS" Value="FMMF" /]
[MapItem Content="ACOMP_APOS" Value="FAAF" /]
[MapItem Content="DIRECT" Value="TFFF" /]
[/Map]
```

Mnemonic is defining which type of synonym will be linked to the Verb Entry - (NO for No synonym).
It can be for example a Movement Direction , a Complement , an Attribute , a Position , a combination Movement Direction Complement ,or anything you want (OTHER)
If OTHER the corresponding pattern must be coded in SIML Files.

```
[Pattern]
[Item]XXXXXCOMP synonymtext[/Item]
[/Pattern]
```

When Synonym Flag is set for verb , Synonym value is configured in Maps **synonym_movement_en** and **synonym_movement_fr** dépending on the language used

```
  [MapItem Content="photograph" Value="a picture" /]
[MapItem Content="film" Value="a video" /]
[MapItem Content="turn off" Value="stopped" /]
[MapItem Content="turn on" Value="started" /]
[MapItem Content="stop" Value="stopped" /]
[MapItem Content="bow down" Value="your body down" /]
[MapItem Content="lift" Value="up" /]
```

```

Coherence between synonym type configured for the verb and synonym value isn't controlled and it's up to you to do it - for example if type is POS , you must be sure that the synonym value is a valid Position Entry.
```

Complement
Maps **verb_complement_fr** and **verb_complement_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Complement]

```
[MapItem Content="the head" Value="head" /]
[MapItem Content="your head" Value="head" /]
[MapItem Content="head" Value="%F HEAD" /]
[MapItem Content="your camera" Value="cam" /]
[MapItem Content="your cam" Value="cam" /]
[MapItem Content="the camera" Value="cam" /]
[MapItem Content="the cam" Value="cam" /]
[MapItem Content="cam" Value="%F CAMERA" /]
[MapItem Content="your arms" Value="arms" /]
[MapItem Content="the arms" Value="arms" /]
[MapItem Content="arms" Value="%F ARMS" /]
[MapItem Content="your left arm" Value="left arm" /]
[MapItem Content="the left arm" Value="left arm" /]
[MapItem Content="left arm" Value="%F ARM_LEFT" /]
[MapItem Content="your right arm" Value="right arm" /]
[MapItem Content="the right arm" Value="right arm" /]
[MapItem Content="right arm" Value="%F ARM_RIGHT" /]
[MapItem Content="your arm" Value="arm" /]
[MapItem Content="the arm" Value="arm" /]
[MapItem Content="an arm" Value="arm" /]
[MapItem Content="arm" Value="%F ARM" /]
[MapItem Content="the last video" Value="%T LAST_FILE VIDEO" /]
```

If the first character of Value is not % , entry will be linked to the value.
For example "your head" will be equivalent to "head" which is the main entry.
On a main entry content
Word 1
Character 1 = %
Character 2 specify if or not a parameter will be present in Word 3 - T True if present , F False if not.
Word 3 if present is a parameter value - this value can be retrieved when building Bot Response Message by value of User Variable **Complementargument**
Word 2 is a mnemonic link to the Map **verb_complement_group** in which are defined for the mnemonic link entry global configuration parameters (valid for every language used).

```
[MapItem Content="0|HEAD" Value="%F LRUD LRUD WITHHEAD HEAD HEAD" /]
[MapItem Content="0|CAMERA_RECORD" Value="%F NO NO WITHCAM CAMERA_RECORD CAMERA_RECORD" /]
[MapItem Content="0|CAMERA" Value="%F LRUD LRUD WITHCAM CAMERA CAMERA" /]
[MapItem Content="4|CAMERA" Value="%F NO NO WITHFIXEDCAM CAMERA CAMERA" /]
[MapItem Content="0|HANDS" Value="%F FBUD NO WITH2ARMS HANDS HANDS" /]
[MapItem Content="0|HAND_LEFT" Value="%F FBLUD NO WITH2ARMS HAND HAND_LEFT" /]
[MapItem Content="0|HAND_RIGHT" Value="%F FBRUD NO WITH2ARMS HAND
```

HAND_RIGHT" /]
[MapItem Content="0|HAND" Value="%T FBUD NO WITH2ARMS HAND EMPTY" /]
[MapItem Content="0|ARMS" Value="%F FBUD NO WITH2ARMS ARMS ARMS" /]
[MapItem Content="0|ARM_LEFT" Value="%F FBLUD NO WITH2ARMS ARM ARM_LEFT" /]
[MapItem Content="0|ARM_RIGHT" Value="%F FBRUD NO WITH2ARMS ARM ARM_RIGHT" /]
[MapItem Content="0|ARM" Value="%T FBUD NO WITH2ARMS ARM EMPTY" /]

```

Content key is structure with 2 parts separated by | delimitor :
Part 1 is the robot index or 0 if the configuration is valid for all the robot types - Use of specific robot index is useful to restrict for example the authorized movement directions ( CAMERA is an example)
Part 2 is the previous complement mnemonic link
In the value returned
Word 1 :
Character 1 is %
Character 2 specify if Complement is or not ambigous T if Yes F if No - If Ambigous dialog asking for precision will be automatically initiated with the User - Example ARM is ambigous - We need to know if user is speaking from left or right arm.
Word 2 is used to specify which Movement Directions are Authorized for [Complement] - It can be overwrited at the global Combination Level.
Word 3 is used to specify which Secondary Movement Directions are Authorized for [Complement] -  It can also  be overwrited at the at the global Combination Level.
Word 2 and Word 3 are mnemonic for movement directions authorizations  Linked to b[]movement_autho_model[/b] map.
Word 4 specify the Robot Type authorization for the Complement - mnemonic linked to Robot Type authorization Map.
Word 5 is the Complement AuthoID
Word 6 is the Complement BBID
For example
```
 [MapItem Content="0|CAMERA" Value="%F LRUD LRUD WITHCAM CAMERA CAMERA" /]
]
```

Not ambigous - Primary and secondary movement direction authorized are Left Right Up and Down - Authorized for All Robot type WITHCAM
The BBID and AuthoID are CAMERA
**Attribute**
Maps **complement_attribut_fr** and **complement_attribut_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Attribute]
[Attribute] needs a preceding  [Verb]  [Complement]  or  [Verb]  [Position] structure.
```
 [MapItem Content="from left to right" Value="left to rightT" /]
[MapItem Content="left to right" Value="%F LEFTTORIGHT" /]
[MapItem Content="from the left to the right" Value="left to right" /]
```
If the first character of  Value is not % , entry will be linked to the value.
For example "from left to right" will be equivalent to "left to right" which is the main entry.
On a main entry content
Word 1
Character 1 = %
```

Character 2 specify if or not a parameter will be present in Word 3 - T True if present , F False if not.
Word 3 if present is a parameter value - this value can be retrieved when building Bot Response Message by value of User Variable **Attributeargument**
Word 2 is a mnemonic link to the Map **complement_attribut_group**.
```

[MapItem Content="LEFTTORIGHT" Value="% LEFTTORIGHT" /]
[MapItem Content="RIGHTTOLEFT" Value=" % RIGHTTOLEFT" /]
[MapItem Content="DOWNTOUP" Value="% DOWNTOUP" /]
[MapItem Content="UPTODOWN" Value="% UPTODOWN" /]
[MapItem Content="RECOGNITION" Value="% RECOGNITION" /]
[MapItem Content="FOLLOWEDCOLOR" Value="% FOLLOWEDCOLOR" /]

```
where word1 is % and word 2 is the attribute BBID and authID

Position
Maps **position_complement_fr** and **position_complement_en** are used to configure for the suffixed language (fr and en) the text keywords which will be recognized in User Input messages as a [Position]
```
   [MapItem Content="paused" Value="%FF PAUSE" /]
[MapItem Content="in pause" Value="paused" /]
[MapItem Content="on pause" Value="paused" /]
[MapItem Content="the headstand" Value="%FF HEADSTAND" /]
[MapItem Content="a headstand" Value="the headstand" /]
[MapItem Content="in the position of the pear treee" Value="%FF HEADSTAND2" /]
[MapItem Content="the side splits" Value="%FF SPLITS" /]
[MapItem Content="a side splits" Value="the side splits" /]
[MapItem Content="the splits" Value="the side splits" /]
[MapItem Content="a splits" Value="the side splits" /]
[MapItem Content="some pushups" Value="pushups" /]
[MapItem Content="pushups" Value="%FF PUSHUPS" /]
[MapItem Content="somersaults" Value="somersault" /]
[MapItem Content="somersault" Value="%FF SOMERSAULT" /]

```
 If the first character of  Value is not % , entry will be linked to the value.
For example "on pause" will be equivalent to "paused" which is the main entry.
On a main entry content
Word 1
Character 1 = %
Character 2 specify if or not a parameter will be present in Word 3 - T True if present , F False if not.
Char 3 is reserved for future use
Word 3 if present is a parameter value - this value can be retrieved when building Bot Response Message by value of User Variable **Positionargument**
Word 2  is a mnemonic link to the Map **position_complement_group** in which are defined for the mnemonic link entry global configuration parameters (valid for every language used).
```


[MapItem Content="HEADSTAND2" Value="%F JD HEADSTAND2 HEADSTAND" /]
[MapItem Content="HEADSTAND" Value="%F JD HEADSTAND HEADSTAND" /]
[MapItem Content="SPLITS" Value="%F JD SPLITS SPLITS" /]
[MapItem Content="PUSHUPS" Value="%F JD PUSHUPS PUSHUPS" /]
[MapItem Content="CALIBRATE" Value="%F ALL CALIBRATE CALIBRATE" /]
[MapItem Content="SIT" Value="%F JD SIT SIT" /]
[MapItem Content="STANDUP" Value="%F JD STANDUP STANDUP" /]
[MapItem Content="CROUCH" Value="%F JD CROUCH CROUCH" /]
[MapItem Content="LEAN" Value="%F JD LEAN LEAN" /]
[MapItem Content="STOP" Value="%F ALL STOP STOP" /]
[MapItem Content="START" Value="%F ALL START START" /]
[MapItem Content="SPREAD" Value="%F JD SPREAD EXTEND" /]
[MapItem Content="ONCROSS" Value="%F WITH2ARMS ONCROSS EXTEND" /]
[MapItem Content="PAUSE" Value="%F WITHCAM PAUSE PAUSE" /]
```

```
Word 1 :
Character 1 is %
Character 2 specify if Position is or not ambigous T if Yes F if No - If Ambigous dialog asking for precision will be automatically initiated with the User - Not used in example
Word 2 specify the Robot Type authorization for the Position - mnemonic linked to Robot Type authorization Map.
Word 3 is the Position AuthoID
Word 4 is the Position BBID
For example
```
```
  [MapItem Content="PAUSE" Value="%F WITHCAM PAUSE PAUSE" /]

```
```
Not ambigous - Authorized for All Robot type WITHCAM
The BBID and AuthoID are CAMERA
Direct calls without  a preceding [verb] must be configured in the SIML maps **synonym_global_fr** or **synonym_global_en** depending on the language used.
Each entry authorized in direct call must refererenced in these maps  - value for each entry give the text synonym to use with this direct call (including [verb].
```
```
[MapItem Content="in initial position" Value="put you in initial position" /]
```
```
```

Combination
Authorized combinations in command user input between [Verb][Complement][Attribute] and [Position] must be specified in Map **combination_verb_comp_att_pos**

```
    [MapItem Content="0|SEND|MAIL_MESSAGE|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO EMPTY EMPTY mailuser TO NO EMPTY messdest" /]
[MapItem Content="0|SEND|TWITTER_MESSAGE|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO EMPTY EMPTY twitteruser TO NO EMPTY messdest" /]
[MapItem Content="0|SHOOT|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="0|TAKE|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="fr|DO|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="0|START|CAMERA_RECORD|EMPTY|START" Value="% NO EMPTY EMPTY WITHCAM" /]
[MapItem Content="0|START|CAMERA_RECORD2|EMPTY|START" Value="% NO EMPTY EMPTY WITHCAM" /]
[MapItem Content="0|STOP|CAMERA_RECORD|EMPTY|STOP" Value="% NO EMPTY EMPTY WITHCAM" /]
[MapItem Content="0|TAKE|CAMERA_RECORD1|EMPTY|EMPTY" Value="% DURATIONNEED NO EMPTY WITHCAM" /]
[MapItem Content="0|TAKE|CAMERA_RECORD2|EMPTY|EMPTY" Value="% DURATIONNEED NO EMPTY WITHCAM" /]
[MapItem Content="0|RECORD|CAMERA_RECORD1|EMPTY|EMPTY" Value="% DURATIONNEED NO EMPTY WITHCAM" /]
[MapItem Content="fr|DO|CAMERA_RECORD1|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="0|CENTER|HEAD|EMPTY|EMPTY" Value="% NO NO EMPTY EMPTY" /]
[MapItem Content="0|CENTER|CAMERA|EMPTY|EMPTY" Value="% NO NO EMPTY WITHMOVINGCAM" /]
[MapItem Content="0|MOVE|ARMS|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVE|ARM|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVE|HAND|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVE|HEAD|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVE|CAMERA|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY WITHMOVINGCAM" /]
```

Content key is structured with 5 parts separated by | delimitor :
Part 1 is the language code (fr , en , ...) or 0 if the configuration is valid for all the language Part 2 is verb AuthID - Part 3 is Complement AuthID - Part 4 is Attribute AuthID and Part 5 is Position AuthID
EMPTY keyword is used for part 2 to 5 when no signifiant.
Value returned :
Word 1 : %

Word 2 : Mnemonic used to specify:
. Whether or not a [Movement Direction] or an [Attribute] is Mandatory for the combination - If Mandatory a Dialog will be initiated with the user to get the mandatory value.
. Whether or Not a [Duration] can be present or is mandatory in the User Input.
. Whether or Not a required [Movement Direction] must be the last phrase element(for example "go 3 steps forward" is authorized but not "go forward 3 steps")
. Whether or Not a required [attribute] must be the last phrase element
. Whether or Not a [Free attribute] can be present or is mandatory in the User Input.
If a [Free attribute] is authorized or mandatory , Word 6 to word 10 must be filled in the MapItem value.
When a [Free attribute] is present in the User Input , which type of decoding and control is done on the [Free attribute] structure : No decoding and control or DECODE0 or DECODE1 or DECODE2.
When DECODE0 , DECODE1 or DECODE2 [Free attribute] structure is [preposition] [article] [attribute text] and structure decoding is done.
With DECODE1 and DECODE2 , control is done on authorized [preposition] and authorized [article] dépending on control type mnemonics in word 7 and word 8 (See after)
DECODE2 add a control on [attribute text] value which must be present in SIML Map **genre_name**
Possible Mnemonics for Word 2 are configured in **combination_autho_model** Map.

```

[MapItem Content="NO" Value="FFFFFFF" /]
[MapItem Content="DURATIONNEED" Value="FFFFMFF" /]
[MapItem Content="DURATIONAUTHORIZED" Value="FFFFAFF" /]
[MapItem Content="MOVENEEDBUTLAST" Value="TTFFFFF" /]
[MapItem Content="MOVENEED" Value="TFFFFFF" /]
[MapItem Content="ATTNEED" Value="FFTFFFFF" /]
[MapItem Content="ATTNEEDBUTLAST" Value="FFTTFFFF" /]
[MapItem Content="ATTANDFREENEED" Value="FFTFFFMF" /]
[MapItem Content="ATTANDFREENEEDDECODE1" Value="FFTFFFM1" /]
[MapItem Content="ATTANDFREENEEDDECODE2" Value="FFTFFFM2" /]
[MapItem Content="FREENEED" Value="FFFFFMF" /]
[MapItem Content="FREENEEDDECODE1" Value="FFFFFM1" /]
[MapItem Content="FREENEEDDECODE2" Value="FFFFFM2" /]
[MapItem Content="FREEPOSSIBLE" Value="FFFFFAF" /]
[MapItem Content="FREEPOSSIBLEDECODE1" Value="FFFFFA1" /]
[MapItem Content="FREEPOSSIBLEDECODE2" Value="FFFFFA2" /]
[MapItem Content="MOVENEED_DURATION" Value="TFFFMFF" /]
[MapItem Content="ATTNEED_DURATION" Value="FFTFMFF" /]

```

Word 3 is used to specify which Movement Directions are Authorized for the combination - Overwrite what it was configured at Verb and Complement level.
Word 4 is used to specify which Secondary Movement Directions are Authorized for the combination
Word 3 and Word 4 are mnemonic for movement directions authorizations linked to **movement_autho_model** map.
Word 5 specify the Robot Type authorization for the Combination - mnemonic linked to Robot Type authorization Map.
Word 6 to 10 must be filled only if a [Free attribute] is authorized or mandatory.
Word 6 : SIML Map Name in which [attribute text] value must be configured - EMPTY if no control map

Word 7 : Mnemonic used to specify the authorized prepositions in free attribut input
Word 8 : Mnemonic used to specify the authorized articles in free attribute input.
Word 9 : Mnemonic used to specify control type on [attribute text] if no entry found in Control Map(Word 6) or Word 6 = EMPTY.In this release only "NUMERIC" is coded and control that format of [attribute text] is Numeric Integer.
Word 10 : [Message name] to send to the user when a mandatory [Free attribute] is configured and not input by the user.
Message need to be configured in Random Responses SIML File with name [Message name]_language code.
For example if value for [Message name] is MESSDEST , tags [Random Name="Messdest_fr"] and [Random Name="Messdest_en"] must be included in Random Responses SIML File.
Mnemonics for authorized [Preposition] Word 7 are configured in map **autho_preposition**
Authorized prepositions are TO , OF , AS , ON and ABOUT

```

[MapItem Content="NOCONTROL" Value="FFFFFFFFFF" /]
[MapItem Content="ALL" Value="TTTTTTTTTT" /]
[MapItem Content="NO" Value="TFFFFFFFFF" /]
[MapItem Content="TO" Value="FTFFFFFFFF" /]
[MapItem Content="OF" Value="FFTFFFFFFF" /]
[MapItem Content="NOANDOF" Value="TFTFFFFFFF" /]
[MapItem Content="NOANDTO" Value="TTFFFFFFFF" /]
[MapItem Content="NOANDAS" Value="TFFTFFFFFF" /]
[MapItem Content="AS" Value="FFFTFFFFFF" /]
[MapItem Content="NOANDON" Value="TFFFTFFFFF" /]
[MapItem Content="ON" Value="FFFFTFFFFF" /]
[MapItem Content="ONABOUT" Value="FFFFTTFFFF" /]

```

Mnemonics for authorized [Article] Word 8 are configured in map **autho_article** and can be a/an or the or no article or plural form

```

[MapItem Content="NOCONTROL" Value="FFFF" /]
[MapItem Content="ALL" Value="TTTT" /]
[MapItem Content="NO" Value="TFFF" /]
[MapItem Content="AAN" Value="FTFF" /]
[MapItem Content="THE" Value="FFTF" /]
[MapItem Content="PLURAL" Value="FFFT" /]
[MapItem Content="ALLEXCEPTPLURAL" Value="TTTF" /]
[MapItem Content="AANTHEPLURAL" Value="FTTT" /]
[MapItem Content="AANTHE" Value="FTTF" /]
[MapItem Content="NOTHEPLURAL" Value="TFTT" /]

```

```

```
[MapItem Content="0|SWING|HEAD|EMPTY|EMPTY" Value="% ATTNEEDBUTLAST LR EMPTY EMPTY" /]
```
```

In this Example , Combination SWING|HEAD is authorized - If No [attribute] is input , It will request to the User to complete with an [Attribute] value.
In this case , after user input the combination SWING|HEAD|Input Attribute will be controlled again.
If [Movement Direction] is input , Authorized Movement Directions are only Left and Right
Robot Type Authorization are the same as those specified in Complement Map - Keyword EMPTY

```
 [MapItem Content="0|SING|EMPTY|EMPTY|EMPTY" Value="% FREENEED NO EMPTY EMPTY song NOCONTROL NOCONTROL EMPTY EMPTY" /]
```

In this example SING without complement attribute or position is authorized. [Free attribute] input is required, If No [Free Attribute] is input , It will request to the User to complete with a [FreeAttribute] value.
No control is done in free attribute input on preposition and article (Keyword NOCONTROL).
Free attribute input will be song title and will be controlled in SIML map song.
Robot Type Authorization are the same as those specified in Complement Map - Keyword EMPTY.

Free Attribute
Structure of [free Attribute] can be either [Attribute text] or [preposition] [article] [attribute text]
As seen above When [free Attribute] is possible or mandatory for a [Verb][Complement] [Attribute][Position] entry in Combination SIML Map,
More informations need to be set in the entry content mainly which type of decoding and control is done on the [Free attribute] structure - No control or DECODE0 or DECODE1 or DECODE2
With No control - [Attribute text] = [Free Attribute] - No decoding is done on [free Attribute] content.
With DECODE0 DECODE1 and DECODE2 , [free Attribute] content is analysed and decoded as syntactic structure [preposition] [article] [attribute text]
Word 6 of entry content is used to specify SIML Map Name in which [attribute text] value can be configured.
Word 9 specify control format on [attribute text] if no entry in the previous control map or if no map specified (Value "EMPTY" in Word 6)
With DECODE2 type mnemonic keyword found for the corresponding [attribute text] entry in map genre_name replace as key [attribute text].
Structure of these SIML Maps must respect following rules :
. Map entry is structured as "Language code|Attribute Text" where language code can be 0 if entry is applicable to all languages.
. 3 Entry content types can be used.
If the first character of Value Content is not % , entry will be linked to the value.
Otherwise the second character after % specify the entry content type - 1 2 or 3
Type 1
Word 2 can be EMPTY , USER or BOT
Case EMPTY value, you find in word 3 the signifiant content value

Case USER or BOT value, word 3 specify name of a user or bot variable.
Signifiant content value for the entry will be found in the same map with key 0|value of user or bot variable.

```
[Map Name="mailuser"]
[MapItem Content="en|my wife" Value="%1 USER wife" /]
[MapItem Content="0|Jean luc" Value="%1 EMPTY jeanlucben29@gmail.com" /]
[MapItem Content="0|Helen" Value="%1 EMPTY helenben29@gmail.com" /]
[MapItem Content="0|XXXXXCURRENTUSERNAME" Value="%1 USER Name" /]
```

In this example signifiant content value is mail adress - entry en|my wife links to 0|value of User variable Wife
If value of User variable wife is Helen - Entry 0|Helen will be searched to get the mail adddress.
Type 2
Word2 can be EMPTY or a mnemonic which will be used as suffix preceded by _ in Bot response message building "workflag" variable structure.

```
[Map Name="Stopcomplement"]
[MapItem Content="fr|de chanter" Value="%2 SING" /]
```

```
[MapItem Content="fr|de danser" Value="%2 DANSE" /]
[MapItem Content="fr|de parler" Value="%2 SOUND" /]
[MapItem Content="en|singing" Value="fr|de chanter" /]
[MapItem Content="en|dansing" Value="fr|de danser" /]
[MapItem Content="en|talking" Value="fr|de parler" /]
```

Type 3
Word 2 is a mnemonic key value
Word 3 is the name of a SIML Map where signifiant content information linked to the text attribute value are stored with the key value of word 2.
Content value structure of this map can be whatever you want (You will need to interprate it in the corresponding SIML Code)
For example

```
[Map Name="song"]
[MapItem Content="fr|joyeux anniversaire" Value="0|happy birthday" /]
[MapItem Content="0|happy birthday" Value="%3 happy_birthday mp3_song" /]
[Map Name="mp3_song"]
[MapItem Content="happy_birthday" Value="% Official_Songs 9" /]
[/Map]
```

In mp3_song map you find in the content entry EZB soundboard name and track number.
Some TIPS for using [Free attribute]
If you want to have plural names decoding with DECODE1 or DECODE2 type , you need to create the corresponding entry in maps **name_pluriels_language code**

```
  [Map Name="name_pluriels_en" Reverse="name_pluriels_reversed_en"]
[MapItem Content="nightingales" Value="nightingale" /]
[MapItem Content="lions" Value="lion" /]
[MapItem Content="birds" Value="bird" /]
[MapItem Content="tigers" Value="tiger" /]
```

If so plural form will be decoded in singular form as set in the entry map content en genre will be set to "P"
You can have more control in using DECODE2 Type, but you need to create an entry in map **genre_name** :

```
 [Map Name="genre_name"]
[MapItem Content="fr|rossignol" Value="%MM nightingale" /]
[MapItem Content="fr|lion" Value="%MM lion" /]
[MapItem Content="fr|lionne" Value="%FF lion" /]
```

Key entry is Language code|decode name.
Entry content structure is :
Char 1 : % char 2 : genre of name char 3: genre
Word 2 : mnemonic keyword for entry.
You can see in the example than lionne and lion have the same mnemonic keyword.
This mnemonic keyword is used instead of [attribute Text) in accessing the Control Map configure for [Free Attribute] interpretation.

As example Entries in Map **combination_verb_comp_att_pos**

```
[MapItem Content="fr|DO|CRY|EMPTY|EMPTY" Value="% FREENEEDDECODE2 NO EMPTY EMPTY animalcry OF ALL EMPTY messwhat" /]
[MapItem Content="en|DO|CRY|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO EMPTY EMPTY animalcry OF ALL EMPTY messwhat" /]
```

Note that we are using DECODE1 for english language and DECODE2 for french. Control map is map Animalcry.

```
 [Map Name="animalcry"]
[MapItem Content="0|LION" Value="%3 LION mp3_animalcry" /]
[MapItem Content="0|TIGER" Value="%3 TIGER mp3_animalcry" /]
[/Map]
```

Entries LION and TIGER are used for french and english languages.
In English, key link directly to the [Attribute Text] decoded with DECODE1 type.
In Map genre_name, only entries for french language are set with mnemonic Lion and Tiger, allowing to have the same keys as for english language.
If we decided to use DECODE2 type for english language , corresponding entries will need to be set in map genre_name with en|prefix.
And Plural form entries are set in the corresponding name_pluriels_fr and name_pluriels_en maps.
In this example - following [free text] inputs will be correctly recognized with the same result :
In french : du lion , d'un lion , de la lionne , d'une lionne , des lions , des lionnes
In english : from lion , from the lion , from a lion , from lions , from the lions

EZ Robot Command SIML Framework - Configuring the Bot Response Message

All SIML Codes to configure Bot Response Message is grouped in SIML File **EZ Robot request - Bot Response Message building**
in models XXXXXXXPROCESSEZCOMMAND and XXXXXXXMESSAGEBUILDING
After interpretation of user Input request by the Framework - Variable **Workflag** is set depending the input phrase syntaxic structure after [Verb]
[Verb complement][movement direction] or [Verb complement][movement direction][secondarydirection]
Worfflag is set to [BBID Complement]_[mnemonic directions] where mnemonic directions is a combination of primary and secondary directions if any.
mnemonic is F,B,L,R,U,L for Forward, Backward,Left,Right,Up and Down and can be configued in map movement_direction_global
Example CAMERA_UL - BBID Complement=CAMERA - Direction=Up Secondary Direction=Left
[movement direction] or [movement direction][secondarydirection]
Worfflag is set to [BBID Verb]_[mnemonic directions] where mnemonic directions are F or B.
Left and Right global movement are converted in adapting left and right speeds.
Example TURN_F - BBID Verb=TURN - Direction=Forward
[Verb Complement]
Worfflag is set to [BBID Complement]_[BBID Verb]
Example - ARMS_EXTEND - BBID Complement=ARMS - BBID Verb=EXTEND
[Verb Complement]][Attribute]
Worfflag is set to [BBID Complement]_[BBID Attribute]
Example - HEAD_LEFTTORIGHT - BBID Complement=HEAD - BBID Attribute=LEFTTORIGHT
[Position]
Worfflag is set to POSITION_[BBID Position]
Example - POSITION_SPLITS - BBID Position=SPLITS
[Action speed]
Worfflag is set to SPECIAL_SPEED.
if only **[verb]** Worfflag is set to [BBID Verb].
if a **[free attribute]** is specified, workflag value can be complemented with _[BBID free atribute] when specified in free control map for the free attribute value(Type 2)
Exemple for [Position] STOP
Entry in Authorized combinations map is

```
[MapItem Content="0|STOP|EMPTY|EMPTY|STOP" Value="% FREEPOSSIBLEDECODE1 NO EMPTY EMPTY Stopcomplement NOANDOF NO EMPTY" /]
```

With free control map set to Stopcomplement
In map Stopcomplement you find in each entry value a type 2 (%2) free attribute value

```
[MapItem Content="en|any movement" Value="%2 ALLMOVES" /]
```

[BBID free atribute] is in this case ALLMOVES and workflag is set to POSITION_STOP_ALLMOVES
Map **synonym_action_robot** enable to configure synonyms for a robot index or for all robots (index 0)

```
[Map Name="synonym_action_robot"]
For example entry in this map[Code][MapItem Content="0|POSITION_STOP_ALLMOVES"
Value="% ALLMOVES_STOP" /]
```
and workflag value POSITION_STOP_ALLMOVES will be transformed to ALLMOVES_STOP for every robot types.

```
[MapItem Content="1|CAMERA_CENTER" Value="% HEAD_CENTER" /]
```

For robot index 1 (JD) - workflag value CAMERA_CENTER will be transformed to HEAD_CENTER and EZ Command will be the same for "Center Cam" and "Center Head"
In the main SIML [Switch Var="Workflag"]
Several models for Building EZ Command can be easily managed
Variable aigui is used to select the building message action :
aigui = "" implicit value
Setting Variable EZFrame with frame name
Or Setting Variable EZAction with action name
Or Setting Variable EZScript with script name
Or Setting Variable tempcommand with a complete EZB command or command sequence (each command separate by | separator)
Will build the EZ Builder corresponding Command.
Variables can be set to select the EZB control name for autoposition plugin name and for script manager plugin name for example :

```
[Var Set="Scriptcontrolname"]Script Manager Synbot[/Var]
[Var Set="Frameactioncontrolname"]Auto Position[/Var]]
```

Variable EZvariablename can be set to an EZBuilder variable name and variable EZvariablevalue to the value to set - this EZBuilder variable will be set in the build command to the value to set if not Empty.
Used for example to Set an EZbuilder variable value which can be tested in a launched script - for example in
```
[Case Value="STEPPING_F"][Var Set="EZvariablename"]Nbsteps[/Var][Var
Set="EZvariablevalue"][User Get="Complementactionfree" /][/Var][Var
Set="EZscript"]Steps Forward[/Var][/Case]
```

where the number of steps is set in tne EZ Builder variable Nbsteps.
aigui = "1"
Variable Gotofunction is used to redirect the process to model XXXXXXXPROCESSCOMMAND_[Var Get="Gotofunction" /]
You can in this model build your specific EZ Command Message and use variables values set during input syntaxic analysis.
Variables can be set before redirecting for using in the processing model - for example
```
[Case Value="SING"][Var Set="tempparam"], "ignoreScript"[/Var][Var Set="Notsay"]|[/
Var][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUNDBOARD[/Var][/Case]
```

aigui = "2"
Forward or Reverse EZB command will be sent depending on variable tempdirection value, value of variables wotkspeedleft and workspeedright will be used for Setspeed command , value of worduration variable will be used to generate a sleep command
aigui = "6" and **aigui = "7"**
Can be set in case the command cannot be executed due to device status - example CAMERA_START and camera already active - only text message response is built (no command)
aigui = "8"
Can be set when command is existing but not yet coded - message "I have not yet learned this command" will be sent in the bot response.
aigui = "9"
Unknown command.
Standard text messages are Build in XXXXXXXMESSAGEBUILDING Model , but you can overwrite these messages in setting variable mess1 and mess2 (case value of aigui = 6,7,8,9 where no EZ Builder commands are built) for example

```
[Case Value="CAMERA_RECOGNITION"][Var Set="EZscript"]Start Bing Vision Recognition[/Var][Var Set="mess1"]|Waiting for Bing Vision recognition[/Var][/Case]
```

EZ Robot Commands SIML Framework - New command creation step to step

To better illustrate , below are a step to step example some new commands
Instructions are for English configuration.

Adding Take Photo Command

Think first at the phrase structure -
can be : "photograph" , "take" or "shoot" or "do" "a photo" , and same ("take" or "shoot" or "do") with "photo" , "picture" ,"a picture" , "snapshot" , "a snapshot"
We have 4 verb forms - Do , Shoot , Take and Photograph
Do is already configured - Shoot , Take and Photograph need to be created.
We have a complement with several possible forms : picture , photo , a picture , a photo , a photography , ...
The best way is to create a complement with CAMERA prefix for exemple CAMERA_PHOTO
Lets go
map **movement_verb_1_en**

```
[MapItem Content="do" Value="%F NO DO" /]
[MapItem Content="shoot" Value="%F NO SHOOT" /]
[MapItem Content="take" Value="%F NO TAKE" /]
[MapItem Content="photograph" Value="%F COMP TAKE" /]
```

map **synonym_movement_en**

```
[MapItem Content="photograph" Value="a picture" /]
```

map **movement_verb_group**

```
[MapItem Content="DO" Value="% ACOMP_APOS EMPTY EMPTY NO ALL DO DO" /]
[MapItem Content="TAKE" Value="% MCOMP EMPTY EMPTY NO ALL TAKE TAKE" /]
[MapItem Content="SHOOT" Value="% ACOMP EMPTY EMPTY NO ALL SHOOT SHOOT" /]
```

map **verb_complement_en**

```
[MapItem Content="a snapshot" Value="picture" /]
[MapItem Content="snapshot" Value="picture" /]
[MapItem Content="a photography" Value="picture" /]
[MapItem Content="photography" Value="picture" /]
[MapItem Content="a photo" Value="picture" /]
[MapItem Content="photo" Value="picture" /]
[MapItem Content="a picture" Value="picture" /]
[MapItem Content="picture" Value="% CAMERA_PHOTO" /]
```

map **verb_complement_group**

```
```

[MapItem Content="0|CAMERA_PHOTO" Value="%F NO NO WITHCAM CAMERA_PHOTO CAMERA_PHOTO" /]
```

map **combination_verb_comp_att_pos**

```
[MapItem Content="0|SHOOT|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="0|TAKE|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
[MapItem Content="0|DO|CAMERA_PHOTO|EMPTY|EMPTY" Value="% NO NO EMPTY WITHCAM" /]
```

In EZ Robot request - Bot Response Message building SIML File
Variable Workflag will be set to CAMERA_PHOTO_DO CAMERA_PHOTO_TAKE and CAMERA_PHOTO_SHOOT
We can configure synonyms in map **synonym_action_robot**

```
[MapItem Content="0|CAMERA_PHOTO_SHOOT" Value="% CAMERA_PHOTO_TAKE" /]
[MapItem Content="0|CAMERA_PHOTO_DO" Value="% CAMERA_PHOTO_TAKE" /]
```

And Add in the [Switch Var="Workflag"] code in Bot Response Message building SIML File the following lines
```

[Case Value="CAMERA_PHOTO_TAKE"]
[Var Set="flagcam"][x:EZvar Get="IsCameraActive" /][/Var]
[If Var="flagcam" Value="0"]
[Var Set="Aigui"]7[/Var]
[Var Set="idmesscamstopped"]Camera_inactive_[User Get="Userlanguage" /][/Var]
[Var Set="mess2"][Random Get="" /][/Var]
[/If]
[Else][Var Set="tempcommand"]ControlCommand("Camera", CameraSnapshot)[/Var][/Else]
[/Case]
```

And test first in the Synbot Studio Console then in SynBot Plugin
And It's Working if camera is active.....

In my Robot Pictures Folder :

**Adding Command for verb Trample**
Trample is walking at the same place - so we will generate EZ Builder AutoPosition Action SHIMMY
Lets go
map **movement_verb_1_en**

```
[MapItem Content="trample" Value="%F NO TRAMPLE" /]
```
```

```

map **movement_verb_group**

```
[MapItem Content="TRAMPLE" Value="% DIRECT EMPTY EMPTY NO ALL TRAMPLE SHIMMY" /]
```

map **combination_verb_comp_att_pos**[/code]

```
[MapItem Content="0|TRAMPLE|EMPTY|EMPTY|EMPTY" Value="% NO NO EMPTY EMPTY" /]
```

Next add code in Bot Response Message building SIML File for value of WorkFlag SHIMMY

```
[Case Value="SHIMMY"][Var Set="EZaction"]Shimmy[/Var][/Case]
```

And test first in the Synbot Studio Console then in SynBot Plugin
And It's Working

Adding Commands to manage volume of sound
A little bit more tricky , We will add some commands to manage EZB sound volume :
Lower , decrease , reduce , lessen sound
Increase , augment , raise sound
Increase , augment , raise sound to the max , to a [volume level]
push sound to maximum , to a [volume level]
Lower , decrease , reduce , lessen sound to the min, to zero , to a [volume level]
Cut sound
Put , set sound to the max , to the mean , to zero , to the min , to a [volume level]
We will Set [Verb] [Complement] [FreeAttribute] structure with Up and Down [Movement Direction]
So we first create entry for a new Complement SOUND
adding keywords for sound in map verb_complement_en

```
[Map Name="verb_complement_en"]
[MapItem Content="the volume of sound" Value="sound" /]
[MapItem Content="your volume of sound" Value="sound" /]
[MapItem Content="the volume" Value="sound" /]
[MapItem Content="your volume" Value="sound" /]
[MapItem Content="the sound" Value="sound" /]
[MapItem Content="your sound" Value="sound" /]
[MapItem Content="volume" Value="sound" /]
[MapItem Content="sound" Value="%F SOUND" /]
```

and in map verb_complement_group

```
[MapItem Content="0|SOUND" Value="%F UD NO ALL SOUND SOUND" /]
```
```

We will use 3 maps to control free attribute values - one for global , one for max and one for min

```
 [Map Name="Volumecomplement"]
[MapItem Content="en|maximum" Value="%2 EMPTY MAX" /]
[MapItem Content="en|max" Value="en|maximum" /]
[MapItem Content="en|highest" Value="en|maximum" /]
[MapItem Content="en|fullest" Value="en|maximum" /]
[MapItem Content="en|utmost" Value="en|maximum" /]
[MapItem Content="en|minimum" Value="%2 EMPTY MIN" /]
[MapItem Content="en|min" Value="en|minimum" /]
[MapItem Content="en|lowest" Value="en|minimum" /]
[MapItem Content="en|zero" Value="en|minimum" /]
[MapItem Content="en|mean" Value="%2 EMPTY MEAN" /]
[MapItem Content="en|average" Value="en|mean" /]
[MapItem Content="en|medium" Value="en|mean" /]
[MapItem Content="fr|maximum" Value="%2 EMPTY MAX" /]
[MapItem Content="fr|max" Value="fr|maximum" /]
[MapItem Content="fr|fond" Value="fr|maximum" /]
[MapItem Content="fr|plein tube" Value="fr|maximum" /]
[MapItem Content="fr|minimum" Value="%2 EMPTY MIN" /]
[MapItem Content="fr|mini" Value="fr|minimum" /]
[MapItem Content="fr|plus bas" Value="fr|minimum" /]
[MapItem Content="fr|zero" Value="fr|minimum" /]
[MapItem Content="fr|medium" Value="%2 EMPTY MEAN" /]
[/Map]
[Map Name="Volumemaxcomplement"]
[MapItem Content="en|maximum" Value="%2 EMPTY MAX" /]
[MapItem Content="en|max" Value="en|maximum" /]
[MapItem Content="en|highest" Value="en|maximum" /]
[MapItem Content="en|fullest" Value="en|maximum" /]
[MapItem Content="en|utmost" Value="en|maximum" /]
[MapItem Content="fr|maximum" Value="%2 EMPTY MAX" /]
[MapItem Content="fr|max" Value="fr|maximum" /]
[MapItem Content="fr|fond" Value="fr|maximum" /]
[MapItem Content="fr|plein tube" Value="fr|maximum" /]
[MapItem Content="en|mean" Value="%2 EMPTY MEAN" /]
[MapItem Content="en|average" Value="en|mean" /]
[MapItem Content="en|medium" Value="en|mean" /]
[MapItem Content="fr|medium" Value="%2 EMPTY MEAN" /]
[/Map]
[Map Name="Volumemincomplement"]
[MapItem Content="en|minimum" Value="%2 EMPTY MIN" /]
[MapItem Content="en|min" Value="en|minimum" /]
[MapItem Content="en|lowest" Value="en|minimum" /]
[MapItem Content="en|zero" Value="en|minimum" /]
[MapItem Content="fr|minimum" Value="%2 EMPTY MIN" /]
[MapItem Content="fr|mini" Value="fr|minimum" /]
[MapItem Content="fr|plus bas" Value="fr|minimum" /]
[MapItem Content="fr|zero" Value="fr|minimum" /]
[MapItem Content="en|mean" Value="%2 EMPTY MEAN" /]
[MapItem Content="en|average" Value="en|mean" /]
[MapItem Content="en|medium" Value="en|mean" /]
```

[MapItem Content="fr|medium" Value="%2 EMPTY MEAN" /]
[/Map]
```

Note in the map content value "%2 EMPTY MAX" the third word which will be used in building command.
We need to be careful , because lower verb is already used with a synonym defined in map synonym_movement_en

```
[MapItem Content="LOWER" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVE MOVE" /]
```

```
[MapItem Content="lower" Value="down" /]
```

As the AuthoID MOVE is shared with other verbs , we need first to change it to a not used MOVEDOWN

```
[MapItem Content="LOWER" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVEDOWN MOVE" /]
```

and to add in combination map

```
[MapItem Content="0|MOVEDOWN|ARMS|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVEDOWN|ARM|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVEDOWN|HAND|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVEDOWN|HEAD|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY EMPTY" /]
[MapItem Content="0|MOVEDOWN|CAMERA|EMPTY|EMPTY" Value="% MOVENEED EMPTY EMPTY WITHMOVINGCAM" /]
MapItem Content="0|MOVEDOWN|SOUND|EMPTY|EMPTY" Value="% MOVENEEDFREEPOSSIBLEDECODE1 EMPTY EMPTY EMPTY Volumemincomplement TO NOTHE NUMERIC EMPTY" /]
```

Note that in order to be able to understood "Lower sound to minimum" , we add FREEPOSSIBLE and control of free attribute in map Volumemincomplement
Note also than in order to be able to understand "Lower Sound to 67", we add a NUMERIC Control Keyword which means that if Free attribute value is not found in map Volumemincomplement
A numeric Control will be done to validate or not the free attribute value
Note the option DECODE1 used and the TO mnemonic for preposition authorization flags word and the NOTHE mnemonic for article authorization flag.
"Lower sound TO min" and "Lower sound TO THE min" will be accepted and will return "min" as free attribute value after decoding.
in map movement_verb1_en , we need to create entries for the new verbs

```

```
[MapItem Content="decrease" Value="%F DIR DECREASE" /]
[MapItem Content="reduce" Value="decrease" /]
[MapItem Content="lessen" Value="decrease" /]
[MapItem Content="increase" Value="%F DIR INCREASE" /]
[MapItem Content="raise" Value="increase" /]
[MapItem Content="augment" Value="increase" /]
```

and to create synonyms in map synonym_movement_en

```
[MapItem Content="increase" Value="up" /]
[MapItem Content="decrease" Value="down" /]
```

and to add entries in map movement_verb_group

```
[MapItem Content="INCREASE" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVEDOWN
MOVE" /]
[MapItem Content="DECREASE" Value="% MCOMP EMPTY EMPTY NO EMPTY MOVEUP
MOVE" /]
```

We add entry for MOVEUP Authoid in combination map with control of free attribute in map Volumemaxcomplement

```
[MapItem Content="0|MOVEUP|SOUND|EMPTY|EMPTY" Value="%
MOVENEEDFREEPOSSIBLEDECODE1 EMPTY EMPTY EMPTY Volumemaxcomplement TO
NOTHE NUMERIC EMPTY" /]
```

We create next verb entries in map movement_verb1_en for verbs put (Already existing) set , push and cut

```
[MapItem Content="set" Value="%F NO SET" /]
[MapItem Content="push" Value="%F NO PUSH" /]
[MapItem Content="cut" Value="%F NO CUT" /]
```

and in map movement_verb_group

```
[MapItem Content="SET" Value="% MCOMP EMPTY EMPTY NO EMPTY SET SET" /]
[MapItem Content="PUSH" Value="% MCOMP EMPTY EMPTY NO EMPTY PUSH PUSH" /]
[MapItem Content="CUT" Value="% MCOMP EMPTY EMPTY NO EMPTY CUT CUT" /]
```

We add entries to combination map for new verbs Authoid

```
[MapItem Content="0|PUT|SOUND|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO
EMPTY EMPTY Volumecomplement TO NOTHE NUMERIC EMPTY" /]
```

```
[MapItem Content="0|SET|SOUND|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO EMPTY EMPTY Volumecomplement TO NOTHE NUMERIC EMPTY" /]
[MapItem Content="0|PUSH|SOUND|EMPTY|EMPTY" Value="% FREENEEDDECODE1 NO EMPTY EMPTY Volumemaxcomplement TO NOTHE NUMERIC EMPTY" /]
[MapItem Content="0|CUT|SOUND|EMPTY|EMPTY" Value="% NO EMPTY EMPTY EMPTY" /]
```

As action will be similar , we gather in map synonym_action_robot the workflag value generated into one

```
[MapItem Content="0|SOUND_SET" Value="% SOUNDVOLUME" /]
[MapItem Content="0|SOUND_PUT" Value="% SOUNDVOLUME" /]
[MapItem Content="0|SOUND_PUSH" Value="% SOUNDVOLUME" /]
```

Now we need to create code for sound increasing and decreasing in EZ Robot request - Bot Response Message building SIML file with 3 new Workflag values : SOUND_U and SOUND_D and SOUNDVOLUME

```
[Case Value="SOUND_U"][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUND_CHANGE UP [User Get="Complementactionfree" /][/Var][/Case]
[Case Value="SOUND_D"][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUND_CHANGE DOWN [User Get="Complementactionfree" /][/Var][/Case]
[Case Value="SOUND_CUT"][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUND_CHANGE SET 0[/Var][/Case]
[Case Value="SOUNDVOLUME"][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUND_CHANGE SET [User Get="Complementactionfree" /][/Var][/Case]
```

With redirections to model XXXXXXXPROCESSCOMMAND_SOUND_CHANGE first parameter (UP , DOWN , SET) and second parameter (MAX , MIN , MEAN or number)
Note that Variable Complementactionfree is automatically set in the framework either to the third word in entry content of map used to control the free attribute value or if entry not found in map to the decoded value of free attribute.

```
[Model]
[Pattern]XXXXXXXPROCESSCOMMAND_SOUND_CHANGE [MAP:KEYVOLUME] $[/Pattern]
[Response xml:space="preserve"]
[Think]
[!--Get first current volume of EZB Sound --]

[Var Set="currentvolume"][x:EZcmd]Print(GetVolume())[/x:EZcmd][/Var]
[If Var="currentvolume" Value=][Var Set="currentvolume"]0[/Var][/If]

[Var Set="upordown"][Match /][/Var]
[Var Set="volumelevel"][Match At="2" /][/Var]
[If Var="upordown" Value="DOWN"][Var Set="newvolume"][Math][Var Get="currentvolume" /]-[Bot Get="Deltasoundvolume" /][/Math][/Var][Var Set="newvolumebis"]200[/Var][/If]
```

```
[If Var="upordown" Value="UP"][Var Set="newvolume"][Math][Var Get="currentvolume" /]
+[Bot Get="Deltasoundvolume" /][/Math][/Var][Var Set="newvolumebis"]0[/Var][/If]
[Switch Var="volumelevel"]
[Case Value="MAX"][Var Set="newvolumebis"]200[/Var][/Case]
[Case Value="MIN"][Var Set="newvolumebis"]0[/Var][/Case]
[Case Value="MEAN"][Var Set="newvolumebis"]80[/Var][/Case]
[Case Value=][Var Set="Nolevel"]1[/Var][/Case]
[Default][Var Set="newvolumebis"][Var Get="volumelevel" /][/Var][/Default]

[/Switch]
[Switch Var="Temperror"]
[Case Value=""]
[If Var="Nolevel" Value="1"][Var Set="newvolumebis"][Var Get="newvolume" /][/Var][/If]
[If Var="newvolumebis" LT="0"][Var Set="newvolumebis"]0[/Var][/If]
[If Var="newvolumebis" GT="200"][Var Set="newvolumebis"]200[/Var][/If]
[User Set="bot_event_response"][Var Get="mess1" /]§Setvolume([Var
Get="newvolumebis" /])§[/User]
[/Case]
[Default]
[!--Case trying to Up to a value lower than current volume or Down to a value greater than
current volume--]
[/Default]
[/Switch]
[/Think]
[/Response]
[/Model]
```

Where we first get the current EZB Sound volume , next compute the new volume to set depending on UP and Down parameter
next test the second parameter if present, and finally build the response message with the EZB Command Setvolume.
We will add some more commands :
Sing Less loud and Sing Louder
Speak Less loud and Speak louder
Here we will use structure [Verb] [Attribute]
We first create entries in map complement_attribut_en

```
[MapItem Content="louder" Value="%T SOUNDVOLUME UP" /]
[MapItem Content="less loud" Value="%T SOUNDVOLUME DOWN" /]
```

We create SOUNDVOLUME entry in map complement_attribut_group

```
[MapItem Content="SOUNDVOLUME" Value="% SOUNDVOLUME1" /]
```

As speak verb is not existing we create it in maps movement_verb_1_en and in movement_verb_group

```
[MapItem Content="speak" Value="%F NO SPEAK" /]
```

```
[MapItem Content="SPEAK" Value="% DIRECT EMPTY EMPTY NO ALL SPEAK SPEAK" /]
```

We add next the combination map entries in map combination_verb_comp_att_pos
Note we add also a combination entry for Speak without attribute to enable the "speak" input - As ATTNEED - Bot will ask precision to the user

```
[MapItem Content="0|SING|EMPTY|SOUNDVOLUME1|EMPTY" Value="% ATTNEED NO EMPTY EMPTY" /]
[MapItem Content="0|SPEAK|EMPTY|SOUNDVOLUME1|EMPTY" Value="% ATTNEED NO EMPTY EMPTY" /]
[MapItem Content="0|SPEAK|EMPTY|EMPTY|EMPTY" Value="% ATTNEED NO EMPTY EMPTY" /]
```

We add code for workflag SOUNDVOLUME1 in Bot Response Message building SIML file.

```
[Case Value="SOUNDVOLUME1"][Var Set="Aigui"]1[/Var][Var Set="Gotofunction"]SOUND_CHANGE [User Get="attributeargument" /][/Var][/Case]
```

Note that variable attributeargument is automatically set with the third word of entry in map complement_attribut_en (UP OR DOWN)
And finally , we want our bot to answer to question about the level of sound volume - For example What's your sound volume , give me the volume of your sound , ...
Just add entry values in map direct_question_object_en

```
[Map Name="direct_question_object_en"]
[MapItem Content="your sound volume" Value="%M SOUNDVOLUME" /]
[MapItem Content="the volume of your sound" Value="%M SOUNDVOLUME" /]

[MapItem Content="volume of your sound" Value="%M SOUNDVOLUME" /]
```

And create code with SOUNDVOLUME keyword in SIML file EZ Robot Request - Queries

```
[Model]
[Pattern]
[Item]XXXXXQUERYEZZZ SOUNDVOLUME[/Item]
[/Pattern]
[Response xml:space="preserve"]
[Think]
[Switch User="Userlanguage"]
[Case Value="fr"][Var Think:Set="Libresponse"]Mon volume sonore est actuellement de [/Var][/Case]
[Case Value="en"][Var Think:Set="Libresponse"]My Sound volume is actually to [/Var][/Case]
[/Switch]
```

[User Think:Set="bot_event_response"][Var Get="Libresponse" /]
[x:EZcmd]Print(GetVolume())[/x:EZcmd][/User]
[/Think]
[/Response]
[/Model]
[Model]
```

Everything is working fine ....
```

Improving vocabulary and commands regognized in the SIML EZ Framework.
Integrating SIX , ROLLI and AdventureBOT specific Commands.
Some New Tags Adaptators are in Progress or are needing to be refactored and adapted to multi language French and English :
Facebook Messenger Input stream
Geonames searching
WeatherUnderground Searching
Date and time SIML models
Calculator models
And in priority **Advanced Learning Models - with the ability to give Learning capabilities to the Bot** - For example Learn to the Bot on family
Learn than Lionel is my son
Learn than Julie is my daughter
Learn than Karine is my daughter
Learn than John is my Son
And give the bot abilities to answer correctly to user input as :
Who are my children ?
Who are my sons ?
Who are my daughters ?
Who is Julie in my family ?