

# IVR Designer Guide

---

## Index

- [1. Introduction](#)
- [2. Login](#)
- [3. Beginning with simple configurations](#)
  - [3.1 Configuration types](#)
  - [3.2 Creating new configurations](#)
  - [3.3 Editing configurations](#)
- [4. Services: Assigning configurations to numbers](#)
- [5. IVR Designer](#)
  - [5.1 Nodes and transitions](#)
  - [5.2 Node manipulation](#)
  - [5.3 Transition editing](#)
  - [5.4 Maps](#)
  - [5.6 Menu actions and node types](#)
    - [5.6.1 File operations](#)
    - [5.6.2 Basic nodes](#)
      - [5.6.2.1 Audio Node](#)
      - [5.6.2.2 Input Node](#)
      - [5.6.2.3 ASR Node](#)
      - [5.6.2.4 RecordFile Node](#)
      - [5.6.2.5 Connect Node](#)
      - [5.6.2.6 Dial Node](#)
      - [5.6.2.7 PlayNumber Node](#)
      - [5.6.2.8 List Node](#)
    - [5.6.3 Advanced nodes](#)
      - [5.6.3.1 ACD Node](#)
      - [5.6.3.2 Mail Node](#)
      - [5.6.3.3 SMS Node](#)
    - [5.6.4 Flow control nodes](#)
      - [5.6.4.1 Bifurcator Node](#)
      - [5.6.4.2 Javascript Node](#)
      - [5.6.4.3 Parse Node](#)
      - [5.6.4.4 Evaluation Node](#)
      - [5.6.4.5 SetVariable Node](#)
      - [5.6.4.6 Pause Node](#)
    - [5.6.5 Development tools menu](#)

[5.6.5.1 Testing maps](#)

[5.6.5.2 View source code](#)

[5.6.5.3 Grid](#)

[5.6.6 Variables](#)

[6. System Requirements](#)

[6.1 Small](#)

[6.2 Medium](#)

[6.3 Large](#)

---

About us

Interactive Powers, SL (EUR)

Calle Magallanes, 13 5º Izq

28015 Madrid (Spain)

Interactive Powers, LLC (USA)

2320 Ponce de Leon Blvd.

Coral Gables, FL 33134 (United States of America)

Website:

<http://www.ivrpowers.com>

Contact us:

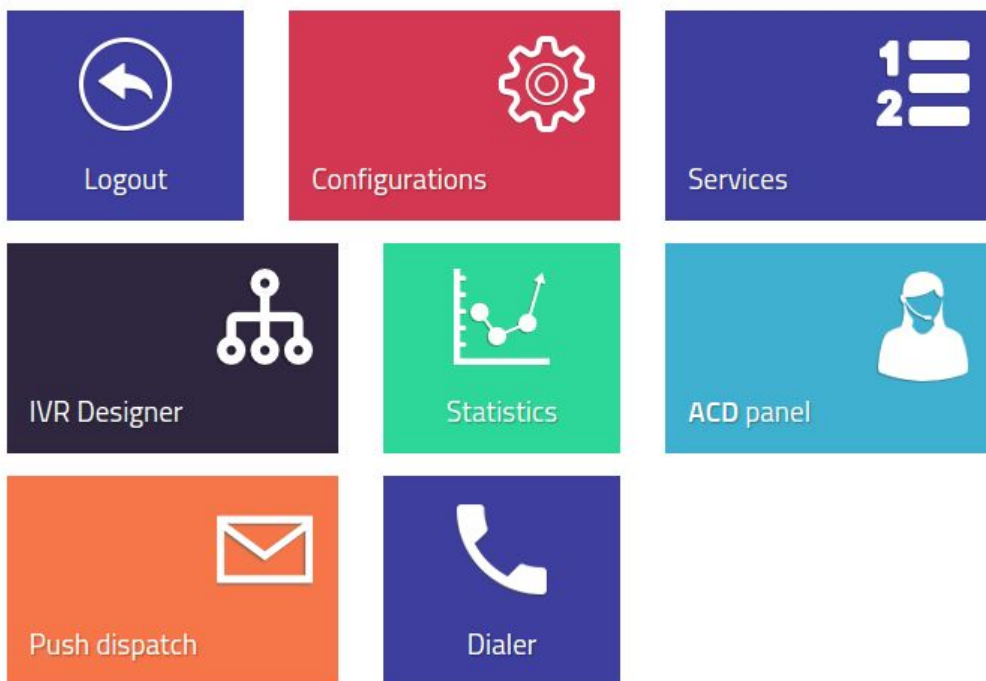
[info@ivrpowers.com](mailto:info@ivrpowers.com)

# 1.Introduction

This user manual describes the usage of this online suite that allows total configuration and management of phone numbers and services, as well as access to additional tools, for a top assistance for your customers, covering all possible communication needs that may arise.

It's possible to perform simple and easy configurations for basic needs. Also it's possible to design and program complex IVRs or voice applications, without the need of advanced programming skills.

With this suite anyone can configure and/or design the behavior of a phone service number, or to modify any aspect in real time in an easy way.



## 2. Login

Ask our sales team for your account login. Different users can get access to different sections depending on roles assigned. With a single logon, your users can access the allowed tools inside the portal.

## 3. Beginning with simple configurations



In configuration section you can define simple service behaviors.

For example:

- Customize your Welcome message
- Send the calls to different destinations
- If the caller calls from a specific region send it to a specific agent
- Route your calls depending of day of week, day of month, etc
- Balance your traffic calls based on percentages

## 3.1 Configuration types

**SIMPLE CONFIGURATIONS:** A simple configuration allows to set up the simplest service: choose the welcoming audio, the destination number where the call will be delivered and the “whisper” that destination will listen to know beforehand the type of call to be answered.

**MULTIPLE CONFIGURATIONS:** It allows to choose the welcome audio and the “whisper”. Call will be sent to up to 3 different destinations. If the first destination does not attend the call (busy or not answering), the call is switched to the second destination and so on.

**SCHEDULE CONFIGURATION:** It allows to configure the line depending on a specific schedule. Outside the configured schedule, the call is not attended and will be hunged up. Besides choosing the start and ending period for the configuration, we can filter by days of the week day within the selected period.

**ORIGIN CONFIGURATION:** With this configuration we can create rules to discriminate the call routing depending on the caller number prefix. It is useful if we want all callers with a prefix belonging to a specific province to be attended with a different language and by operators that speak such language.

**BALANCING CONFIGURATION:** With this option we can choose the percentages of incoming calls that we want to be attended by each destination phone. Thus, an operator can answer, for example 80% of the calls and another 20%, depending on how we wish to distribute the workload on them.

## 3.2 Creating new configurations

Click on “New Configuration” button to create a new Configuration. This configuration can be used in several services (numbers):



Choose the type of configuration desired:

A light blue form titled 'NEW CONFIGURATION' with the subtitle 'Routing Type'. On the right side, there is a dropdown menu with a green header 'Select a configuration' and a list of options: 'Simple', 'Multiple', 'Schedule', 'Origin', and 'Balanced'. Below the list is a white box with the text 'Select a configuration' and a downward arrow.

And complete the form of the particular configuration. In this example we are using Multiple configuration:

A light blue form titled 'NEW CONFIGURATION' with the subtitle 'ROUTING MÚLTIPLE'. The form contains several fields: 'Type' (Multiple), 'Name' (empty), 'Description' (empty), 'Welcome speech' (test 44), 'Record call' (Record checkbox), 'Destination 1' (Ej: 917885051), 'Destination 2' (Ej: 917885052), and 'Destination 3' (Ej: 917885053). There are also buttons for '+', a play icon, 'Cancel', and 'Save'.

### 3.3 Editing configurations

From the Configuration panel, we can view a listing with all configurations previously created, assigned to a service (numbering), or not.

We can use the search option to filter current configurations. To view and/or modify any of them click on it.

If that configuration is already associated to a service, the changes made on it will take effect immediately.

From that menu, we can eliminate the configuration. But we can only delete it if it isn't associated to any numbering, to avoid that the service is non-configured.

## 4. Services: Assigning configurations to numbers



In **Services** section we can view all our services and numbers and define which configuration use with each service.

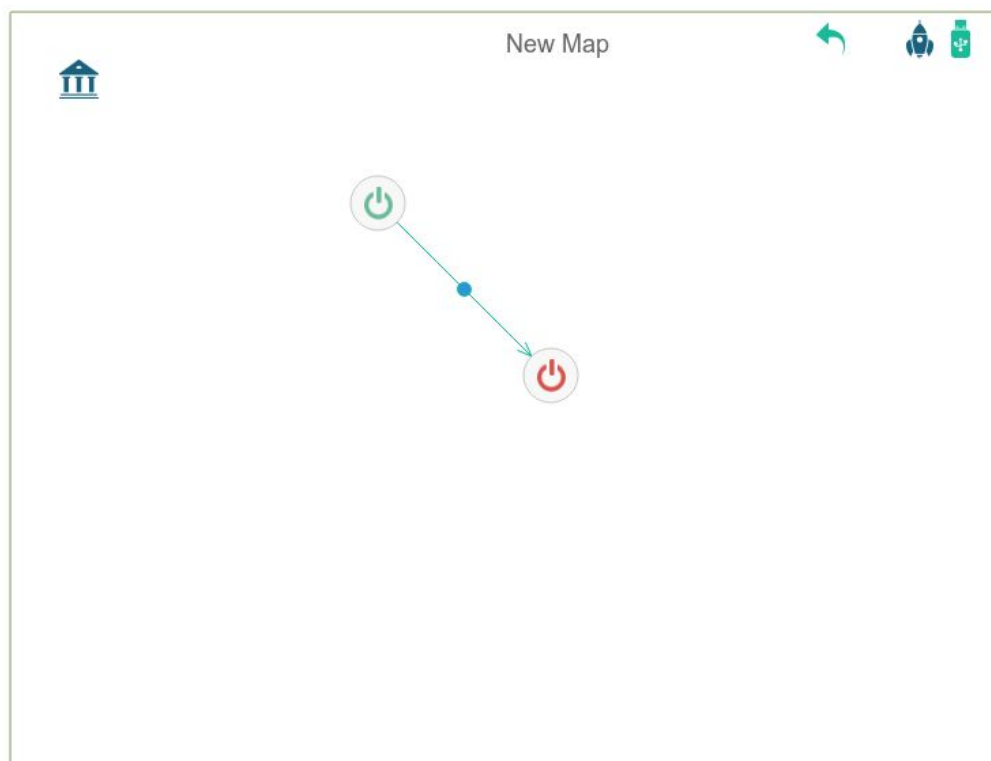
To assign or modify a configuration for each number, click on the number and assign the configuration desired.

## 5. IVR Designer



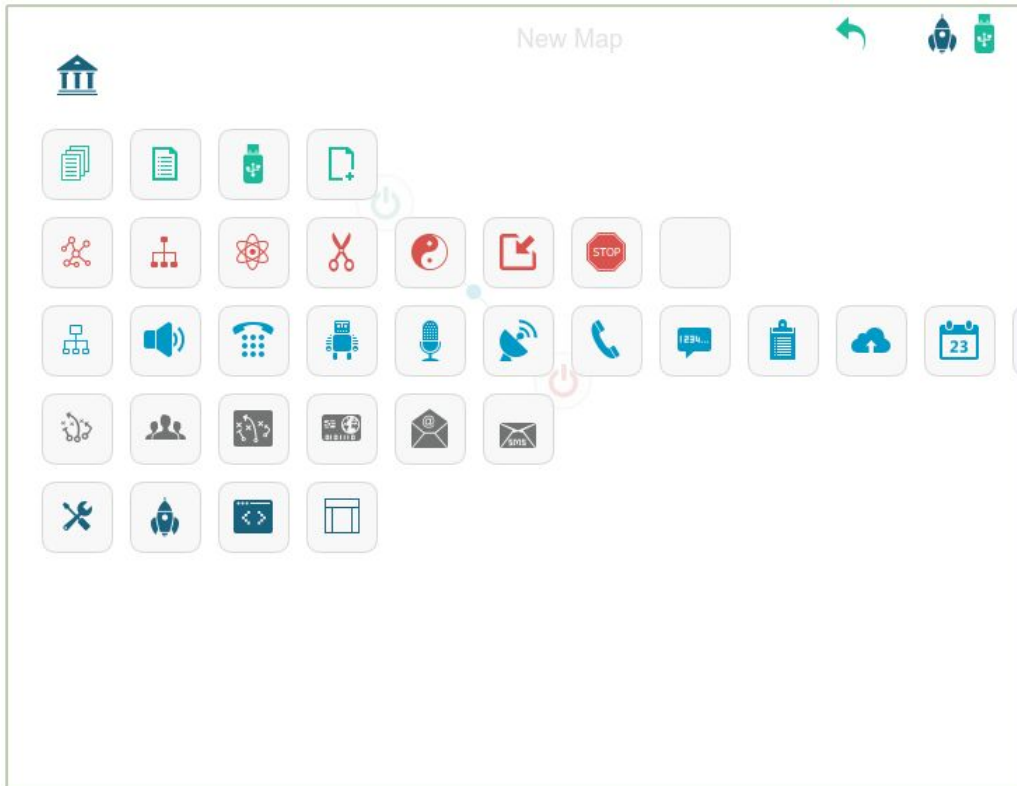
Designer tool allows the development of IVR applications without any programming skills.

The main element of the Designer is the Canvas where you can design your applications (also called maps). Maps are built only with 2 types of elements: **Nodes** and **Transitions**. Changes made on the maps (once saved) have an immediate effect on the behaviour of the new calls entering the system.







On the left side of the canvas we can find the HOME icon where we can select nodes to add to the map.



## 5.4 Maps

Maps can define an application behavior depending on which nodes and transitions are defined.

	The call always starts at the <b>Start</b> node
	The call always ends at the <b>End</b> node

The maps must comply the following simple rules:

- A Map must have a unique **Start** initial mode. This node cannot have incoming transitions and can only have one outgoing transition.
- A map must have an **End** final node. This node can have several incoming transitions. Outgoing transitions will be executed when call is finished (for example, call a Webservice with some data collected during the call).
- All nodes must be reachable






## 5.6 Menu actions and node types

### 5.6.1 File operations

	File Operations
---	-----------------









In this section we can upload maps, save or create new maps.

	Load	Loads a map previously saved
	Save	Saves a Map. You can save with the same name or “Save as” with a new name, creating a new copy of the map
	New	Creates a new Map


## 5.6.2 Basic nodes

	Basic Nodes
---	-------------

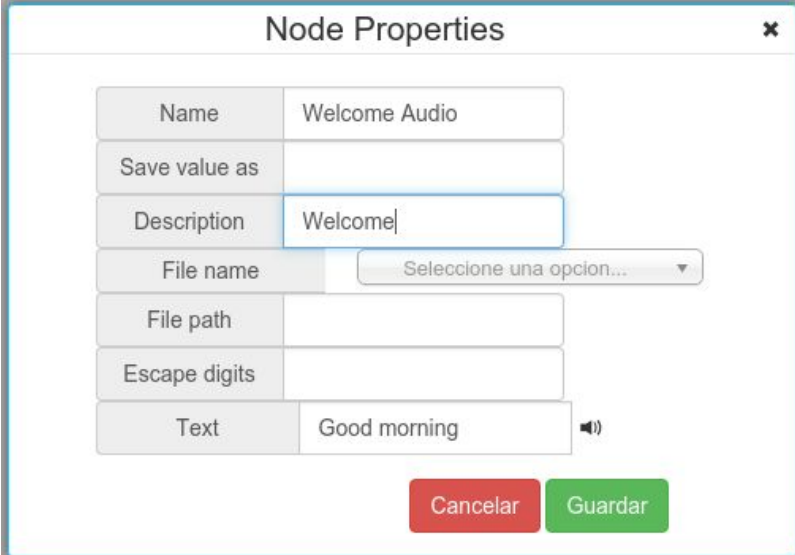
This section explains the basic nodes that allow to build a simple IVR: Audio Recording, Recordings, Text-to-Speech, Speech-Recognition, etc.

	Audio Node
	Input Node
	ASR Node
	RecordFile Node
	Connect Node
	Dial Node
	PlayNumber Node
	List Node


### 5.6.2.1 Audio Node

	<b>Audio Node</b>
<b>Description</b>	This is one of the most important and common node in Designer. It allows the playback of audio files.
<b>Output</b>	This node generate no outputs
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> This node generates no output.</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>File Name:</b> Audio file to play. Its a selector field. Audio files can be uploaded from Resources section.</p> <p><b>File Path:</b> is needed only if used outside the web application, so it can play audio files using local routes for the server, wherever it is executed.</p> <p><b>Escape Digits:</b> Define the keys used to interrupt audio (0-9, #, *)</p> <p><b>Text:</b> Text to generate with TTS engine. If <b>File Name</b> is defined this field has no effect</p>

In following example we find an audio node configured to play the TTS “Good morning”




The screenshot shows a 'Node Properties' dialog box with the following fields:


- Name:** Welcome Audio
- Save value as:** (empty)
- Description:** Welcome|
- File name:** Seleccione una opcion...
- File path:** (empty)
- Escape digits:** (empty)
- Text:** Good morning 

Buttons at the bottom: Cancelar (red), Guardar (green).


### 5.6.2.2 Input Node

	<b>Input Node</b>
<b>Description</b>	Allows to receive the information introduced by the dialpad (DTMF). Users can introduce any sequence of characters with the phone: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, #, *
<b>Output</b>	The DTMF sequence dialed by the user
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Saves the keys typed into a variable. This variable can be read later in another point of the map. For example: if this field has the value "AGE", this variable can be referenced later as {AGE}.</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Audio file:</b> File to be played before waiting for the clicking by the user. For example: "Click 1 to speak with the Commercial Dept., click 2 to speak to the Legal Dept." or "Introduce your age".</p> <p><b>Audio route:</b> Only applicable to offline versions of the platform</p> <p><b>Bargein:</b> It specifies if we want the audio of the presentation to be interrupted or not.</p> <p><b>Text:</b> Text to speech if there is no real audio. No interrupted with keypad action.</p> <p><b>Maximum digits:</b> Maximum number of digits that the user can introduce</p> <p><b>Minimum digits:</b> Minimum number of digits that the user can introduce. If the minimum number of characters is not observed, the process is repeated.</p> <p><b>Timeout:</b> In seconds, maximum time that we will wait for the insertion of digits by the user. 5 seconds by default.</p> <p><b>Repetitions:</b> Number of times that the process is repeated if the user does not introduce the minimum number of digits. 1 by default.</p> <p><b>Error audio:</b> Audio file to be played in the case that the user does not introduce the minimum number of digits. For example "You haven't introduced the 4 characters o your PIN"</p>

### 5.6.2.3 ASR Node


	<b>ASR Node</b>
<b>Description</b>	Speech Recognition. The recognition works based on the grammar selected.
<b>Output</b>	The recognized String
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Saves the recognized string in a variable</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Grammar:</b> Grammar to use for the recognition. Some of the available grammars are:</p> <p>digits: Recognizes digits from 0 to 9          yes-no: Recognizes yes or no</p> <p><b>Audio file:</b> File to be played before waiting for the user's voice. For example: "Say 1 to speak with the Commercial Dept., say 2 to speak with the Legal Dept." or "Say your age".</p> <p><b>Error audio:</b> Audio file to be played if the user doesn't say anything or what has been said is not recognizable. For example "We couldn't understand you, speak slowly and clearly"</p> <p><b>Text:</b> Text to speech in the case of not having a real audio. Non-interruptible with keypad action.</p>

### 5.6.2.4 RecordFile Node

	<b>RecordFile Node</b>
<b>Description</b>	<p>Allows the user to save an audio file.</p> <p>Normally it's needed when the data required from the user is too complex to be collected with keypad tones or through ASR. With this node you can save the audio and can be processed later by back office staff.</p> <p>Its use is common for:</p> <ul style="list-style-type: none"> <li>● Request a postal address</li> <li>● Name and last name</li> <li>● Comments, suggestions, complains...</li> </ul>
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> This node generates no output</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Record File Name:</b> Name of the generated file. It can include combinations of system variables, such as {_CALLID}, {_CALLER} o {_CALLED} to differentiate the different audios.</p> <p><b>Format:</b> Format of the file to create: wav or gsm (uppercase or lowercase)</p> <p><b>Escape digits:</b> Keys that may interrupt the recording. For example for cases such as: "Click * when you finish"</p> <p><b>Timeout:</b> Maximum time that the recording lasts in seconds</p>




### 5.6.2.5 Connect Node


	<p><b>Connect Node</b></p>
<p><b>Description</b></p>	<p>Make HTTP calls to external services to obtain relevant information for the application.</p> <p>Typical uses for this node:</p> <ul style="list-style-type: none"> <li>• Obtain relevant information regarding a user: Status of a request, account balance, results of an operation, etc</li> <li>• To send information to an external application</li> <li>• Query if a user has access to a certain resource or information, etc.</li> </ul> <p>The node allows to perform GET, POST, DELETE, PUT operations. It allows also to send fixed parameters or parameters based in execution or system variables.</p> <p>The results can be received in 2 formats:</p> <ul style="list-style-type: none"> <li>• As a plain text: variable1=value1[separator]variable2=value2... The separator can be any character, normally: #,  , etc. Based on this separator, the node automatically parses the response and stores independent variables (variable1, variable2, etc.)</li> <li>• As JSON format</li> </ul> <p>If a response is received in plain JSON (without composed objects), the response is parsed and stored as independent variables. For example, JSON String:</p> <pre>{   "name": "John Doe",   "age": 18 }</pre> <p>variables "name" and "age" will be injected in context in order be used in the future. So if we use variable {name} we will get "John Doe" value</p>
<p><b>Output</b></p>	<p>Server response, it may be useful to debug.</p> <p>In case of error:</p> <p>ERROR_HTTP_ERROR          ERROR_NO_RESPONSE          ERROR_EMPTY_RESPONSE</p>
<p><b>Properties</b></p>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Saves the server's response into a variable</p>

	<p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>URL:</b> URL where the resource or service is located</p> <p><b>REST Path:</b> Relative route for the resource in case of a request to a REST resource.</p> <p><b>Method (HTTP or REST):</b> Http method or REST request. Possible values: HTTP or REST.</p> <p><b>REST Operation :</b> GET, POST, PUT, DELETE</p> <p><b>Parameter:</b> chain of parameters under GET (v1=c1&amp;v2=c2...) format. The use of variables within parameters with {variable} format is allowed</p> <p><b>Separator:</b> To parse the response. Only for HTTP classic requests. The REST requests ignore this and parse the response JSON as commented before.</p>
--	--


### 5.6.2.6 Dial Node

	<b>Dial Node</b>
<b>Description</b>	Performs outgoing calls to link the incoming call with an external phone service.
<b>Output</b>	The result of the call in numeric format. The codes correspond to the following states: 0-INIT; 1-RING; 3-BUSY; 4-TIMEOUT; 5-FAILURE; 10-ANSWER; 11-FINISHED; 12-CANCEL; 13-CONGESTION; 14-CHANUNAVAIL; 15-DONTCALL; 16-TORTURE; 17-INVALIDARGS; 18-UNKNOWN
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Save call status into a variable. For example: if call was answered and we use the value "CALL_STATUS" the value 10 will be saved into this variable.</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Destination Number:</b> Destination to call</p> <p><b>Max seconds:</b> Max seconds to try the call</p> <p><b>Call ANI:</b> Caller to be displayed in called phone's display</p>


### 5.6.2.7 PlayNumber Node

	<b>Playnumber Node</b>
<b>Description</b>	<p>Plays voice numbers with natural voice instead of using TTS.</p> <p>It is common if we want to spell a longer number.</p> <p>We can use expressions like 123{value}456. If the value of the variable =9, it will play “1239456”</p>
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Number:</b> String to reproduce (it may contain alphanumeric characters and variables)</p> <p><b>Mode:</b> It may contain one of the following values:</p> <ul style="list-style-type: none"> <li>● DIGITS: Reproduces digits one by one. Ex: 134 (One, three, four)</li> <li>● NUMBER: Reproduces the value as a number. Ex: 134 (One hundred thirty four)</li> <li>● ALPHA: Reproduces characters. Ex: ABC (A, B, C)</li> <li>● DATETIME: Reproduces dates. Ex: 20140731 (July, 31st, 2014)</li> </ul>




### 5.6.2.8 List Node

	<b>List Node</b>
<b>Description</b>	Check if a value exists into a list. Typically used to build white lists, black lists, etc
<b>Output</b>	OK if value exists into the list or KO if not
<b>Properties</b>	<b>Name:</b> Name of this node <b>Save Value as:</b> Not used <b>Description:</b> Description of this node. It's useful to documentate the node functionality.


### 5.6.3 Advanced nodes

	Advanced nodes
---	----------------


This section explains the advanced nodes that allow connect your IVR to external elements like an ACD (Call Center Queue), Mailbox, SMSbox...

	ACD Node
	Mail Node
	SMS Node

### 5.6.3.1 ACD Node


	<b>ACD Node</b>
<b>Description</b>	Send the call to the ACD engine
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>ACD Service:</b> Name of ACD service (i.e: Customer Care). This services can be configured from ACD section in the main menu</p> <p><b>ACD Skill:</b> Name of the Skill (i.e: VIP customers)</p> <p><b>Call ANI:</b> Use to mask the caller number in order to avoid ACD agents the see real one.</p>

### 5.6.3.2 Mail Node

	<b>Mail Node</b>
<b>Description</b>	Sends an email. Audio files can be used as attachment
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Contains:</b> Use this field as a filter to include only matching audio files as attachment. If we use value "NAME", audio files containing string "NAME" will be sent</p> <p><b>Erase:</b> true or false. If true, audio files will be deleted after mail sending</p> <p><b>Folder:</b> Folder where audio files have been recorded i.e: /tmp/audios</p> <p><b>Zip file name:</b> File name of the attachment. i.e: test_{_CALLID}.zip</p> <p><b>Email from (remitent):</b> Sender address</p> <p><b>Email to (destination):</b> Recipient address</p> <p><b>Subject:</b> Subject of the mail</p> <p><b>Body:</b> Body of the mail</p>



### 5.6.3.3 SMS Node







	<b>SMS Node</b>
<b>Description</b>	Sends a SMS to a mobile number
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>From:</b> Sender number in the SMS</p> <p><b>To:</b> Mobile number to send the SMS. Variables can be used ({_CALLER})</p> <p><b>Text:</b> Text to be sent. Variables can be used inside the text. For example: "Thank you for calling from {_CALLER} phone"</p>

## 5.6.4 Flow control nodes


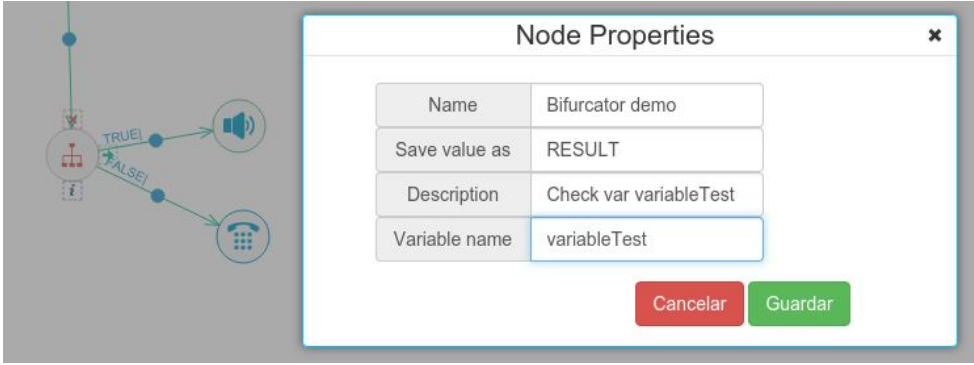
	Flow control nodes
---	--------------------

It contains nodes that allows to control the execution flow in the application: following a certain route or another based on a condition, evaluation of variables and expressions, etc.


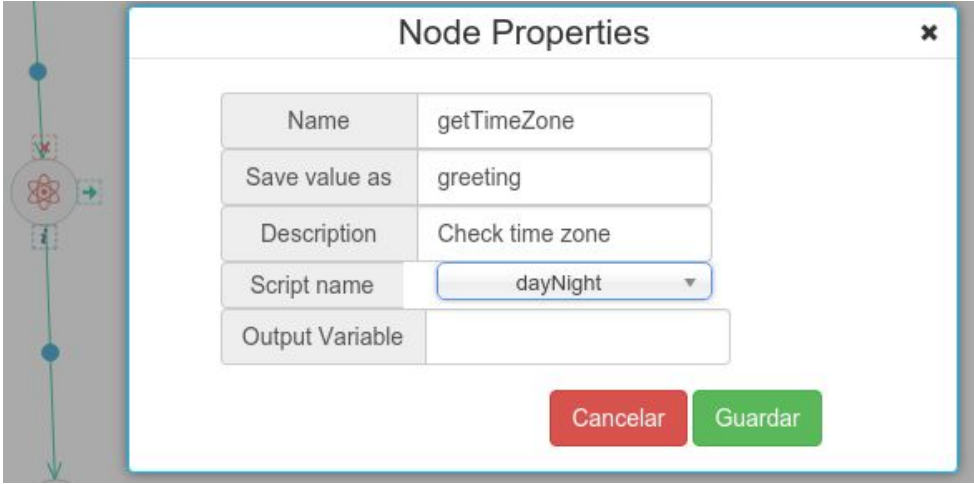
These nodes require basic programming knowledge related to variables, expressions, etc.

	Bifurcation Node
	Javascript Node
	Parse Node
	Evaluation Node
	Set Variable Node
	Pause Node

### 5.6.4.1 Bifurcator Node


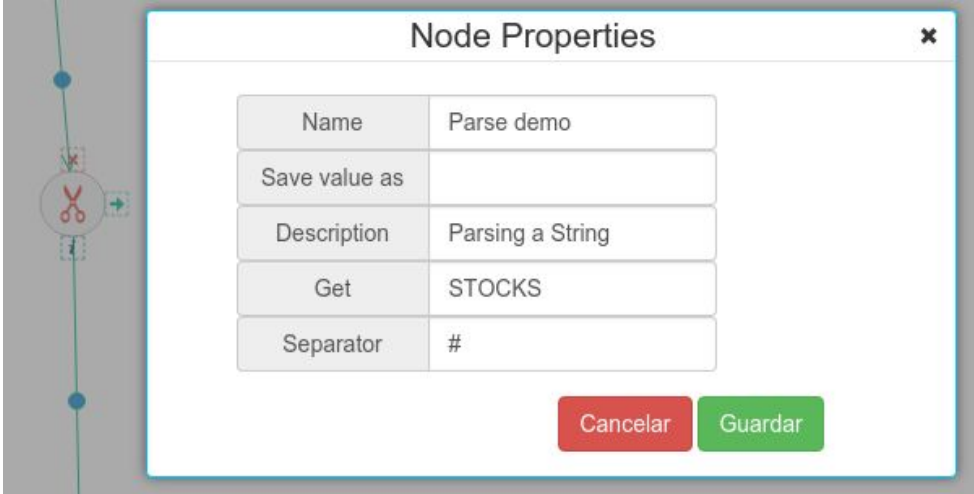
	<p><b>Bifurcation Node</b></p>
<p><b>Description</b></p>	<p>Allows to follow a certain route based on the value of a variable. The name of the variable to evaluate must be inserted in the “Variable Name” field.</p> <p>All possible values of the variable must be written in the outflow transitions of the node.</p> <p>See the following example:</p> <div data-bbox="408 757 1385 1120" style="border: 1px solid gray; padding: 5px;">  </div> <p>In this example if the variable “variableTest” has the “TRUE” value, it will jump to the audio node.</p> <p>If the variable has the “FALSE” value, it will jump to the input node. We can add several conditions. We can also add several conditions into a single transition, for example in the case of TRUE, we could write: TRUE, CORRECT.</p> <p>If the variable does not contain any of the matching values, default transition (blank) will be used. If there is no default transition the call would jump directly to the End node.</p>
<p><b>Output</b></p>	<p>Variable value</p>
<p><b>Properties</b></p>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Save the result into a variable</p> <p><b>Description:</b> Description of this node. It’s useful to documentate the node functionality.</p> <p><b>Variable name:</b> Name of the variable to check</p>

## 5.6.4.2 Javascript Node


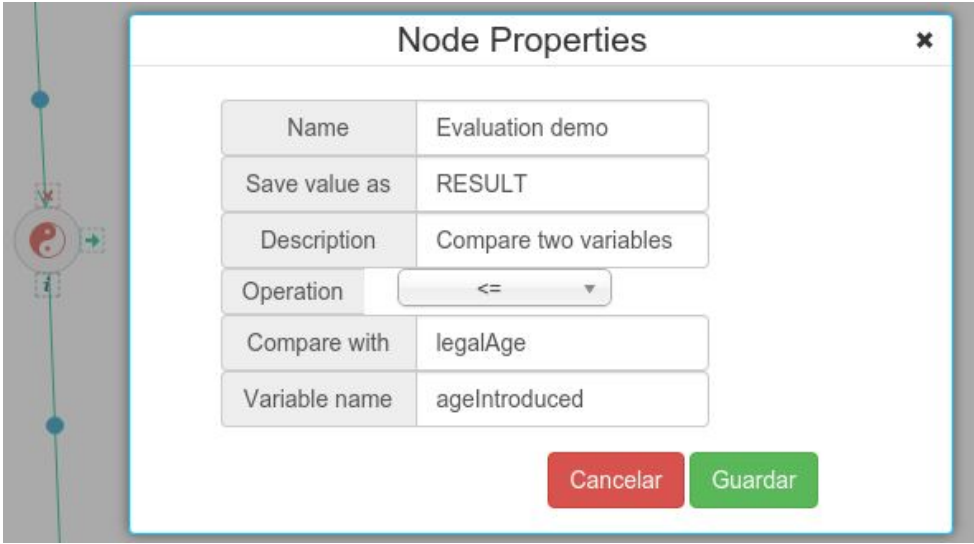
	<b>Javascript Node</b>
<b>Description</b>	<p>Executes JavaScript code to perform operations or complex calculations: Check if an ID card is valid, check a credit card, check if a certain date is valid, mathematical operations, etc.</p> <p>Next we show an example that serves to provide a greeting or another depending on the time of day:</p>  <p>If the time for the execution of the node is the morning (for example from 7:00 to 13:00) the Javascript node will generate the value "GOOD MORNING".</p> <p>Following this transition, the map will lead us to an audio node saying good morning. Similarly for the other 2 cases.</p> <p>The Javascript function used is called "DayNight", and it is defined as:</p> <pre>function checkTime() {   today = new Date();   var nowHour = today.getHours();   if (nowHour&gt;=7 &amp;&amp; nowHour&lt;13)     return "GOOD_MORNING";   else if (nowHour&gt;=13 &amp;&amp; nowHour&lt;20)     return "GOOD_AFTERNOON";   else if (nowHour&gt;=20 &amp;&amp; nowHour&lt;=23)     return "GOOD_NIGHTS";   else if (nowHour&gt;=0 &amp;&amp; nowHour&lt;7)     return "GOOD_NIGHTS";   else return "GOOD"; }</pre>

	<pre>var resultByDefault=checkTime();</pre> <p>The default output of this Javascript node is defined by the variable “resultByDefault” although it can be any other (for this we need to use the “Output var” field). If we wish to store the value to use it in the future, we can use the “Save as” field, in the example the value (GOOD_MORNING, etc) is stored in the “greeting” variable.</p>
<b>Output</b>	The result of the Javascript execution
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It’s useful to documentate the node functionality.</p> <p><b>Script name:</b> Script to be executed.</p> <p><b>Output variable:</b> Output of the script to be used as output for this node. If empty, variable resultByDefault will be used.</p>


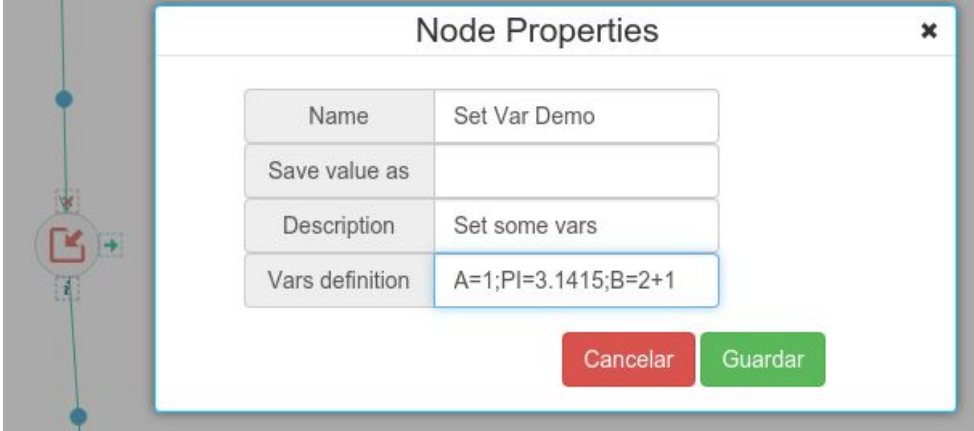
### 5.6.4.3 Parse Node

	<p><b>Parse Node</b></p>
<p><b>Description</b></p>	<p>This node is used to split a composed value into 2 simple values. Often it's used to process the response of an external service.</p> <p>Let's assume that we have a variable "STOCKS" from an external service with the stock values of NASDAQ companies in the following format:</p> <p>AAPL=106.75#GOOG=513.80#AMZN=312.63</p> <p>Let's assume that such value is stored in the "serverResponse" variable: After the execution of this node, all the variables will be injected into conext execution:</p> <p>AAPL=106.75                      GOOG=513.80                      AMZN=312.63</p> 
<p><b>Output</b></p>	<p>No output is generated</p>
<p><b>Properties</b></p>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Get:</b> Variable to be parsed</p> <p><b>Separator:</b> Separator used (i.g: #, ;  ...)</p>

#### 5.6.4.4 Evaluation Node


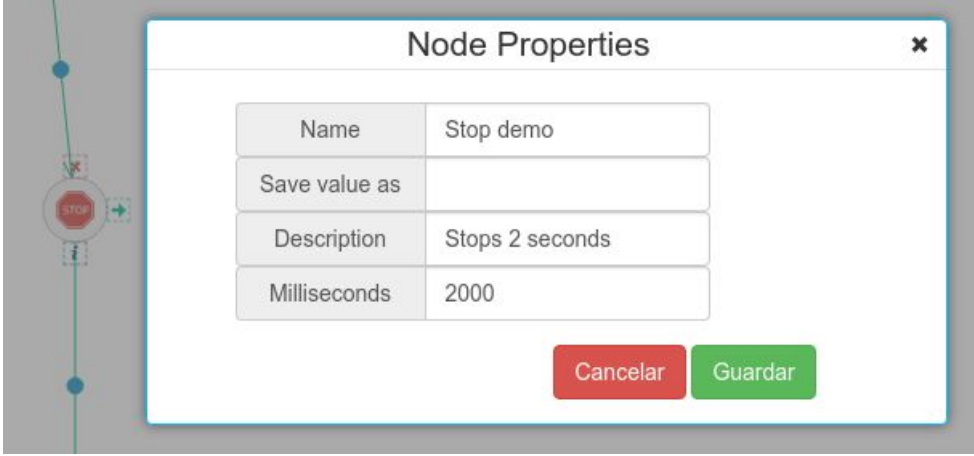
	<h3>Evaluation Node</h3>
<p><b>Description</b></p>	<p>Compares variables in order to make decisions.</p> <p>Assuming that we have 2 variables:  ageIntroduced: Previously given by the user  legalAge: Value of 18</p> <p>Let's assume now that we want to make a voice based on the user's age:</p>  <p>With this configuration, the node performs the following operation:  Is ageIntroduced less than legalAge? If the result is True, the node generates the value TRUE, otherwise, it generates the value FALSE.</p> <p>Uses: Counters, groups, check limits, etc.</p>
<p><b>Output</b></p>	<p>TRUE or FALSE</p>
<p><b>Properties</b></p>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Variable name to save the output value</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Operation:</b> One of following: ==, !=, &gt;, &lt;, &gt;=, &lt;=</p> <p><b>Compare with:</b> Right operand of the expression</p> <p><b>Variable Name:</b> Left operand of the expression</p>

### 5.6.4.5 SetVariable Node

	<b>Set variable Node</b>
<b>Description</b>	<p><b>Sets the value of one or more variables</b></p> <p>Variables can be set using “= ”operator. Use “;” separator if you need to set more than one variable. Expressions with preexisting variables can be also used:</p> <p>A=1</p> <p>or</p> <p>PI=3.14;A=B+1</p> 
<b>Output</b>	No output
<b>Properties</b>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Vars definition:</b> Expression with variables and their values</p>






### 5.6.4.6 Pause Node

	<p><b>Pause Node</b></p>
<p><b>Description</b></p>	<p>Stops the map execution at this point</p> <p>In following example the node forces a 2 seconds pause:</p> 
<p><b>Output</b></p>	<p>No output</p>
<p><b>Properties</b></p>	<p><b>Name:</b> Name of this node</p> <p><b>Save Value as:</b> Not used</p> <p><b>Description:</b> Description of this node. It's useful to documentate the node functionality.</p> <p><b>Milliseconds:</b> Pause time expressed in milliseconds</p>

## 5.6.5 Development tools menu

In this section, you can find tools that will help you with the development: Testing maps, guide to align nodes, check source code, etc

	Testing Maps
	View Source Code
	Grid

### 5.6.5.1 Testing maps

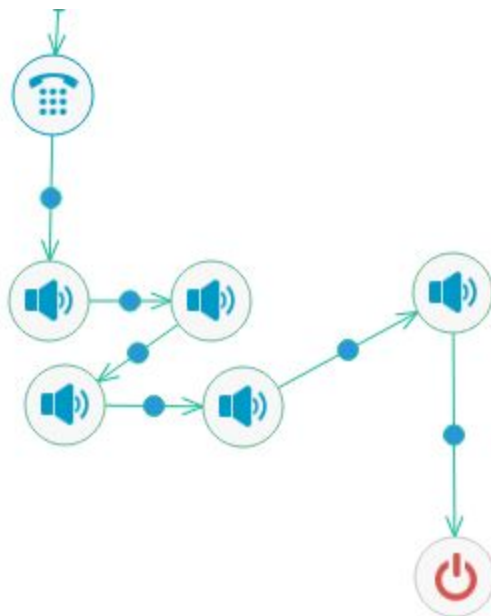
	Testing Maps
---	--------------

It allows to test a map receiving a call from the system playing back the last recorded map.

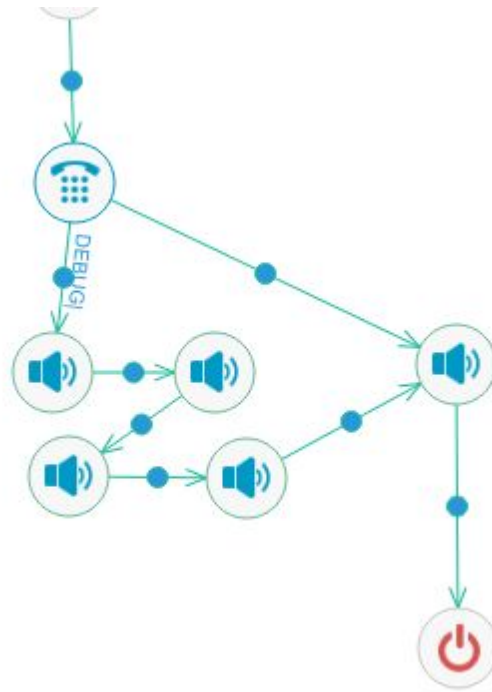
When clicking, a window appears requesting to introduce the phone where we want to receive the test call.

Keep in mind that system variables as `{_CALLED}` o `{_CALLER}` are not available because this is not a real call.


You can use fake values in transitions to avoid run tested sections of a map, for example, in following map we need to test only last audio node, so we want to skip first audio nodes:



So we can change incoming transition with a tag that never will match (for example “DEBUG”, “foo” or whatever) to force the execution to a temporary transition in order to test last audio node:



### 5.6.5.2 View source code

	View Source Code
---	------------------

It allows to verify the source generated by the application internally. It is useful to check that there are no errors.

By clicking in this icon you get information of internal map definition:

```

META centerX 80
META centerY 70
META zoom 1
1 START 8 Inicio posX=152 posY=-12
2 END 0 Fin posX=347 posY=323
3 AUDIO 2 New_3 posX=346 posY=181
4 AUDIO 5 New_4 posX=152 posY=237
5 AUDIO 3 New_5 posX=241 posY=238
6 AUDIO 7 New_6 posX=144 posY=185
7 AUDIO 4 New_7 posX=225 posY=185
8 INPUT 6 New_8 posX=146 posY=82

```

### 5.6.5.3 Grid

	Grid
---	------

It allows to activate vertical and horizontal lines on the background that facilitate the alignment of nodes.

You also can see the FPS (Frame Rate), X/Y coordinates, and other information



### 5.6.6 Variables

Designer can use variables. These variables can be:

**System Variables:** Variables that can be referenced at any moment, defined by the system itself. These variables begin with “\_” and can be referenced as {\_NAME} or On the upper right part we have quick access buttons to save a map, and to test the loaded map (rocket icon).

## 5.1 Nodes and transitions



Maps are built always from 2 types of elements: Nodes and transitions.

**Nodes:** The node element performs a specific action, such as play audio, generate outgoing calls, send sms, etc. Some nodes generates an outgoing value. Each node type performs a specific action and has specific properties which define and/or modify its behaviour. Nodes are explained in section 5.6.2




**Transitions:** A transition is a connection between nodes. A transition always start in a single node and ends in another node or itself. Transitions may be labelled with values to define the conditions to go from one node to another.

## 5.2 Node manipulation

Nodes can be clicked to show available operations:

	
Unselected node	Selected node

Operations over a node:



	Delete this node
	Edit node properties
	Create a new transition starting on this node

## 5.3 Transition editing

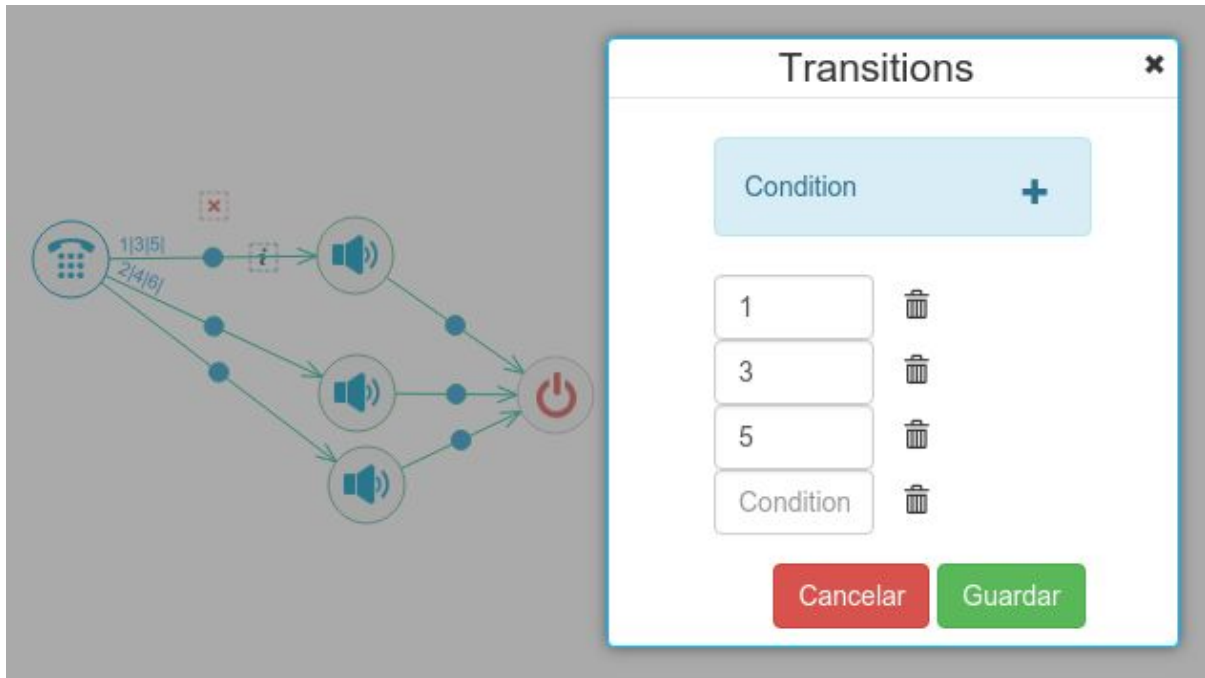
To edit the properties of a transition click on the central vertex:



Operations over a transition:

	Delete this transition
	Edit transition values

We can edit the tags of each transition. In following example we are editing one of the outgoing transitions of the Input node at the left. If DTMF typed is 1, 3 or 5 the call will follow to the upper audio node. If the value typed is 2, 4 or 6 the call will continue through middle audio node. If other key is pressed (default behavior) the last audio node will be played.



#\_NAME#

The available system variables are the following:

{\_CALLER}: The calling number

{\_CALLED}: The called number

{\_CALLID}: Call Id

{\_CHANNEL\_NAME}: Channel Id

{\_CUSTOMER\_ID}: Id of the client owning the graph

{\_ELAPSED\_SECONDS}: Seconds lasted since call started

**META Variables:** Variables defined at the beginning of the graph definition. In META section.

**Execution Variables:** Defined in runtime. Variables can be created using the node's property SAVE\_VALUE\_AS (if that node generates any output) or with the SetVar node. These variables can be accessed as #varName# or {varName}. It is not allowed these variables begin with "\_" since that character is reserved for System variables.



## 6. System Requirements

### 6.1 Small

Capacity: Up to 100 concurrent calls

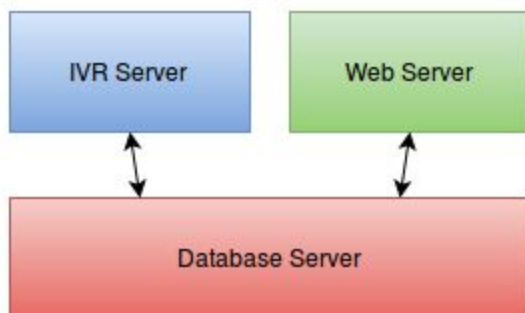
Processor: 1xQuadCore 3.1 ghz

RAM Memory: 8 GB

HDD: 1TB (Raid 1) – 10K RPM (recommended) or above

### 6.2 Medium

Capacity: Up to 500 concurrent calls



#### **IVR Server**

Processor: 2x Intel Xeon E5 cores

RAM Memory: 16 GB

HDD: 1TB

#### **Web Server**

Processor: 1 Intel Xeon E5 core

RAM Memory: 16 GB

HDD: 3TB

#### **Database Server**

Processor: 1 Intel Xeon E5 core

RAM Memory: 16GB

HDD: 5TB

## 6.3 Large

Above 500 concurrent calls

For large configurations it's recommended to make a detailed analysis of the service in order to scale each subsystem

