apstel
DEVELOPING IDEAS

Visual Dialplan 3.3

User Manual

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Apstel. The software described in this document is furnished under a license agreement. It is against the law to copy or reverse engineer the software except as specifically allowed in the license agreement.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Apstel cannot assume responsibility for any existing errors.
Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Trademarks

The Apstel logo and Visual Dialplan are trademarks of Apstel in the U.S. and other countries. Asterisk is registered trademark of Digium Inc. All other marks are the property of their respective owners.

Support

If you have any questions, comments or requests, please contact us at: http://www.apstel.com.

Table of contents

Chapter

1

Introduction

Welcome to Visual Dialplan.

Visual Dialplan for Asterisk is innovative visual modeling platform that enables Asterisk users to create, maintain and test dialplan in a convenient and natural way.

Using point and click user interface, intuitive component editor, predefined sample dialplans and context sensitive help, new dialplan can be built and maintained quickly and easily.

Visual Dialplan brings easy of use and comfort of a modern Rapid Application Development platform into the world of Asterisk without sacrificing functionality of a standard Asterisk dialplan.

Intended audience

This manual is intended for all users involved in planning, developing and maintaining Asterisk dialplan, including Asterisk professionals, consultants, ordinary Asterisk users and Asterisk administrators. The manual assumes that you have a basic understanding of Asterisk PBX.

Chapter summaries

This manual provides information on how to install and use Visual Dialplan for Asterisk. It also provides comprehensive reference manual of Asterisk dialplan applications (components) and functions. The manual contains the following chapters:

- **Chapter 1**, "Introduction" provides an overview of the Visual Dialplan and this manual, and identifies the intended audience.
- **Chapter 2**, "Getting started", provides information about system requirements and gives instructions on how to get started using Visual Dialplan.
- **Chapter 3**, "Visual Dialplan Interface", describes the main window, menus, toolbars, context, variables, macros and Asterisk configuration view, and context and macros design panels.
- **Chapter 4**, "Visual Dialplan Components", lists and provides comprehensive description for each Visual Dialplan component.
- **Chapter 5**, "Visual Dialplan Functions", lists and provides comprehensive description for each Visual Dialplan Function.
- **Chapter 6**, "ASR Grammar", describes the syntax for grammar representation.
- **Chapter 7**, "Visual Dialplan Licensing", describes licensing mechanism.
- **Chapter 8**, "Integration Server", describes Integration Server functionality and how to use Visual Dialplan with Integration Server.

Chapter

2

Getting started

System Requirements

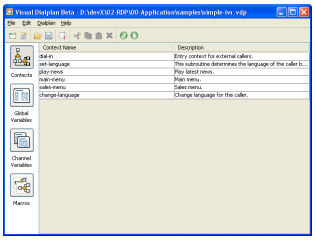
Current version of Visual Dialplan is available for Microsoft Windows and Linux operating systems.

Visual Dialplan requires Java Runtime Environment (JRE) which is bundled with the Visual Dialplan downloadable package and should not be downloaded and installed separately.

First steps with Visual Dialplan

After starting the Visual Dialplan you will be offered with the sample visual dialplans. Visual Dialplan is shipped with several sample dialplans that can be used for educational purposes and may be a good starting point for building your own dialplan.

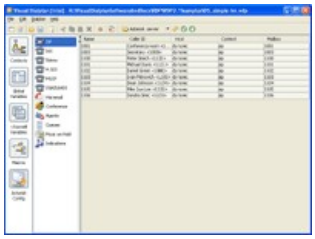
Organize your contexts, macros and variables



When you open existing dialplan or create a new one, you are first presented with the context view.

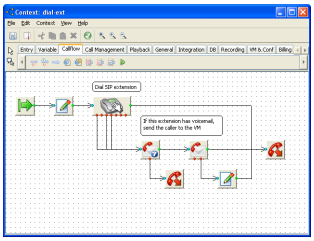
Visual Dialplan enables you to organize and describe contexts, macros, channel and global variables. You can easily switch between context, macro, global variable, channel variable and Asterisk configuration views using the left navigation bar.

Read the server configuration data



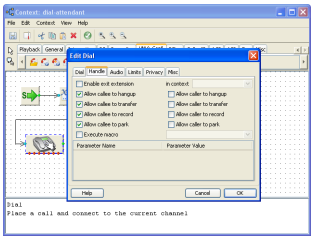
Enable Visual Dialplan to read the Asterisk server configuration data and accommodate its behavior accordingly, by defining the connection to the server at preferences dialog (this connection is defined by default in Visual Dialplan PBX Edition). Then check the Asterisk server configuration data by opening the Asterisk config view.

Visualize your call flow



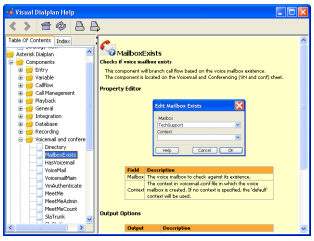
With Visual Dialplan you can organize dialplan building blocks and connect them to create the call flow. Components are organized on the component toolbar in the following categories: Call flow, Call management, Billing, ASR, ACD, etc.

Model don't code



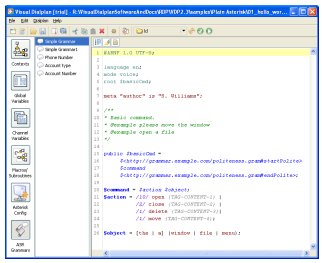
Visual Dialplan provides specialized editors for each component. Just double click the component from the working area and a property editor will appear. You don't have to know syntax and properties for each and every dialplan application. Property editors provide easy and intuitive way for setting these values.

Find information in the local help system



All the information you need to build a dialplan is now available on your desktop, integrated with the Visual Dialplan and presented in a natural way – no need to search the Internet or list the books, just select the component and press F1 to get all the information about that component.

Develop ASR voice applications



Visual Dialplan is perfect tool for building Asterisk dial plan voice applications.

It supports all Asterisk speech recognition related dial plan applications (LumenVox dial plan applications) and comes with feature rich grammar editor.

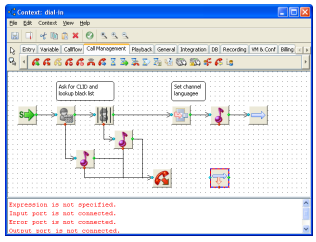
Elastix, AsteriskNOW, trixbox, PIAF ...

Visual Dialplan supports Asterisk PBX but it also supports trixbox, PBX In A Flash, AsteriskNOW, Elastix and all other major distributions.

You can simply create extensions, voicemail boxes and trunks using Elastix (trixbox, PIAF etc.) and then do all the dial plan magic with Visual Dialplan.

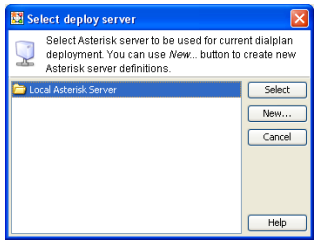
AsteriskNOW, PBX In A Flash, trixbox and Elastix are the property of their respective owners

Validate the dialplan



Check the dialplan before deploying it on the Asterisk box. You can validate a dialplan for errors using the validate toolbar button. Components that are not properly connected or configured will be marked in red and all errors will be listed in the info area.

Deploy the dialplan



When you are ready to deploy the dialplan just click on the Deploy button. Visual Dialplan will generate standard Asterisk dialplan code and deploy it to the Asterisk server. Visual Dialplan will also reload Asterisk server configuration thus making dialplan changes immediate.

Chapter

3

Visual Dialplan Interface

The Visual Dialplan interface is designed to speed up and simplify dialplan development. The interface consists of the following components:

- The Main Window, which is the user's primary access to Visual Dialplan. It consists of the menu, toolbar and the Contexts, Variables, Macros, Asterisk configuration, Scripts/Grammars and Integration Server view.
- Contexts and Macros design panel, which is a graphical view used for contexts and macro visual modeling. Here you can build and edit a context or macro by placing and connecting objects, and defining their properties. It consists of the menu, toolbar, component toolbar with dialplan building objects, visual modeling area and info area.
- Properties editor, which is used for setting up dialplan building object's properties and expressions.

Main Window

The Main Windows is the primary access point to the Visual Dialplan. It consists of the menu, toolbar and the Contexts, Variables, Macros, Asterisk configuration, Scripts/Grammar and Integration Server view.

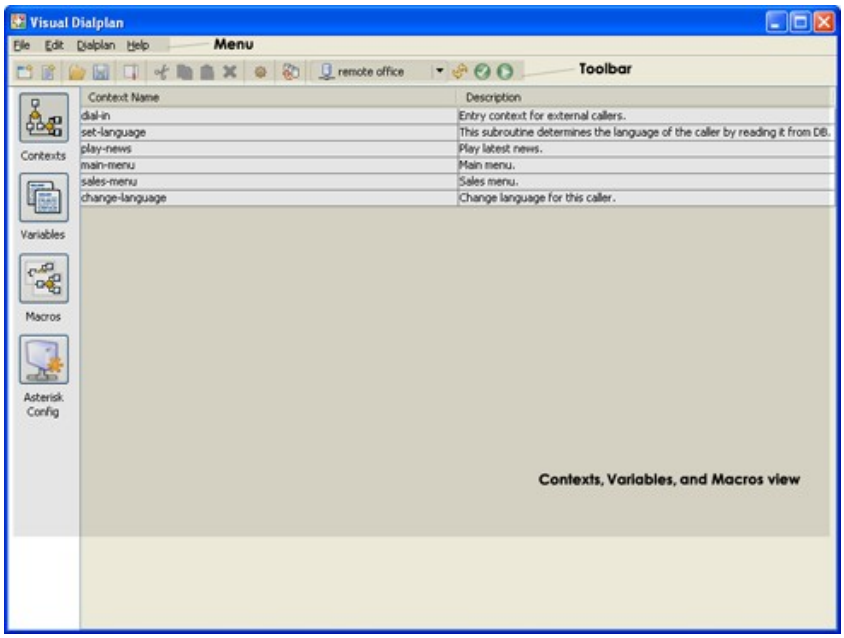


Figure 1 - Main Window

Menu

The following menus are available on the Main Window:

File menu

- New
  Create a new object based on the active view.

  On the example shown on the figure 1 it would be new context but with this option you can also create new Global Variable, new Channel Variable and new Macro.
- New Dialplan
  Create new dialplan.
- Open Dialplan (Import)
  Open new visual dialplan ('vdp' extension) file.

  Note: This functionality is titled 'Import' in Visual Dialplan PBX Edition.
- Save Dialplan
  Save currently opened visual dialplan.
- Save Dialplan as (Export)
  Save currently opened visual dialplan at local file system under new file name.

Note: This functionality is titled ‘Export’ in Visual Dialplan PBX Edition.

- Dialplan properties
  Modify dialplan properties. A dialplan properties dialog enables editing of the following options.
  - Static

    If static is set to no, or omitted, then the pbx_config will rewrite extensions.conf file when extensions are modified.
  - Write protect

    In case the Static property is checked and Write protect property unchecked, user will be enabled to save a dialplan by CLI command 'save dialplan'.
  - Auto fall through

    If this property is checked, then if an extension runs out of things to do, it will terminate the call with BUSY, CONGESTION or HANGUP depending on Asterisk's best guess. This is the default value in Asterisk dialplan v1.4.x and is strongly recommended in Asterisk dialplan v1.2.x.

    If this property is unchecked, then if an extension runs out of things to do, Asterisk will wait for a new extension to be dialed.
  - Clear global variable

    If this property is checked, global variables will be cleared and reparsed on an extensions reload, or Asterisk reload.

    If this property is unchecked, then global variables will persist through reloads, and even if deleted from the extensions.conf or one of its included files, and will remain set to the previous value.
- Close Dialplan
  Close the currently opened dialplan.

- Exit
  Exit Visual Dialplan and close opened dialplan. If there are any unsaved changes in the current dialplan, a confirmation message prompts you to save those changes.

Edit menu

- Cut
  Removes selected objects and moves the objects to the clipboard.
- Copy (ALT-E-C)
  Copies selected objects to the clipboard.
- Paste (ALT-E-P)
  Moves copied objects from the clipboard and create the same objects. Newly created objects will have the same name as the original object with the prefix 'Copy of'.
- Delete (ALT-E-D)
  Deletes selected objects.
- Preferences (ALT-E-P)

  Open Preferences dialog.

Dialplan menu

- Include (ALT-D-I)
  Dialplan include functionality allows user to include existing traditional Asterisk dialplan code.
- Select server (this option is not available in Visual Dialplan PBX Edition)

  Select the Asterisk server you want to build the dialplan for.
- Refresh Configuration

  Read the Asterisk configuration again.
- Validate (ALT-D-V)
  This option will check the dialplan. Components that are not properly connected or configured will be marked in red.
- Deploy (ALT-D-D)
  This option will generate the standard Asterisk dial plan code (extensions.conf code) based on the currently opened visual dialplan and will deploy it to the server.
- Version ... (Change version)
  This option allows you to upgrade/downgrade the dialplan to the different version.

Help menu

- Help Contents (F1)
  Open Visual Dialplan help system.
- Check for updates
  Check if new version of Visual Dialplan is available.
- Purchase
  Purchase Visual Dialplan.
- Register
  Register the trial version of Visual Dialplan.
- Send a question
  Contact us and tell us what you think about Visual Dialplan, we appreciate your feedback very much.
- About (ALT-H-A)
  Software version, copyright and other related information.

Toolbar

The following toolbar icons are available on the Main Window:

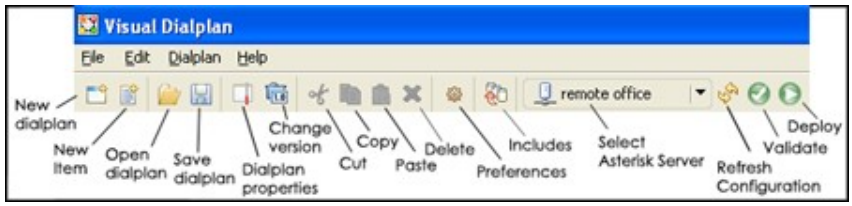


Figure 2 - Main Windows toolbar

- New dialplan
  Create new dialplan.
- New item
  Create a new object based on the active view.

  In the example shown on the figure 1 it would be new context but with this option you can also create new Global Variable, new Channel Variable and new Macro.
- Open dialplan (Import)
  Open new visual dialplan.

  Note: This functionality is titled 'Import' in Visual Dialplan PBX Edition.
- Save dialplan
  Save currently opened visual dialplan.

- Dialplan properties
  Modify dialplan properties. A dialplan properties dialog, as described in previous section, is shown.
- Change version
  This is a very powerful function. It allows you to upgrade/downgrade the dialplan to a different version. Although we try to automate this procedure as much as possible, it can not be completely automated and after the upgrade/downgrade you will probably need to update some of the dialplan components manually. It is highly recommended to make a backup copy before upgrading/downgrading the dialplan because there is no simple way to revert back to the previous state after this action.
- Cut
  Removes selected objects and moves the objects to the clipboard.
- Copy
  Copies selected objects to the clipboard.
- Paste
  Moves copied objects from the clipboard and create the same objects. Newly created objects will have the same name like to origin object with the prefix 'Copy of'.
- Delete
  Deletes selected objects.
- Preferences
  Application Preferences dialog allow you to configure different aspects of the Visual Dialplan application. Using the preferences dialog you can configure the following application parameters:
    - Deployment server parameters
    - General Visual Dialplan parameters
- Include
  Dialplan include functionality allows a user to include existing traditional Asterisk dialplan code.
- Select Asterisk server (this option is not available in Visual Dialplan PBX Edition)

  Under this drop down menu you can select the target Asterisk server you want to build the dialplan for and/or to deploy the built dialplan to. If you have configured several Asterisk servers, the five most recently used servers will be listed under this drop down box.

Important note:
Please note that you will not be able to take full advantage of this product unless you select a target Asterisk server and that way enable Visual Dialplan to read the Asterisk servers configuration and modify its behavior accordingly.

- Refresh configuration

  If you changed the configuration of your target Aseterisk server manually, or using any other GUI on the market, please be sure to let Visual Dialplan know about the configuration change by clicking on "Refresh configuration" button.

- Validate
  This option will check the dialplan. Components that are not properly connected or configured will be marked in red.

- Deploy
  This option will generate the standard Asterisk dial plan code (extensions.conf code) based on the currently opened visual dialplan and deploy it to the server. If the 'Reload dialplan after deploy' option is checked, Visual Dialplan will also reload the Asterisk servers configuration.

Views

The Main Window is the primary access point for Visual Dialplan. From the main window you can access Contexts, Variables, Macros, Asterisk configuration, Scripts/Grammar and Integration Server views.

Contexts view

The Contexts button, shown in Figure 3, is the main access point to all the contexts used in the dialplan. Click on the Contexts button to switch the main window to the context view and to get a list of all contexts.

Any context that didn't pass the validation test will be shown in red, which means those contexts have some errors or warnings and should be fixed before deploying the dialplan.

By double clicking on the context name, the selected context will be opened in the design panel.

Right clicking on the context name will open an additional menu that will allow the user to open the context, create a new context, or cut, copy, and delete the selected context, paste the copied context, or edit context properties.

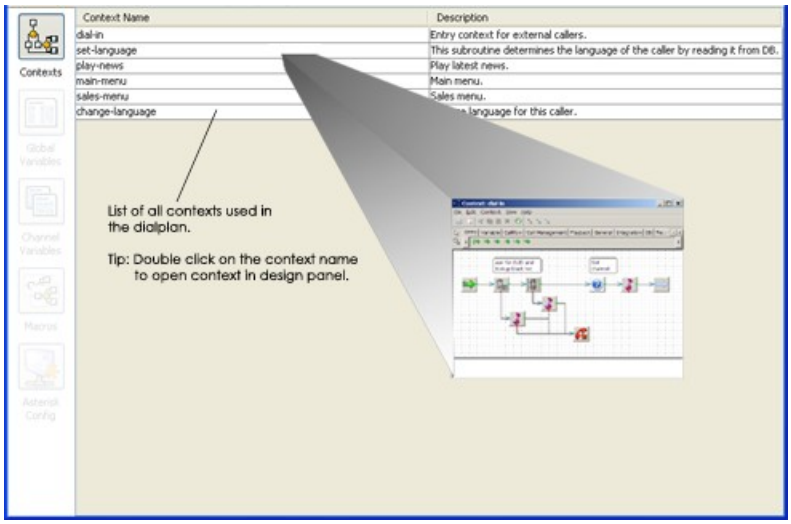


Figure 3 - Contexts view

Variables view

By clicking on the Variables button you will get a list of all global and channel variables used in the dialplan. Double clicking on the variable name will open the property window and will allow you to modify variable properties. Right click will open additional menu: open, new, properties, cut, copy, paste, delete.

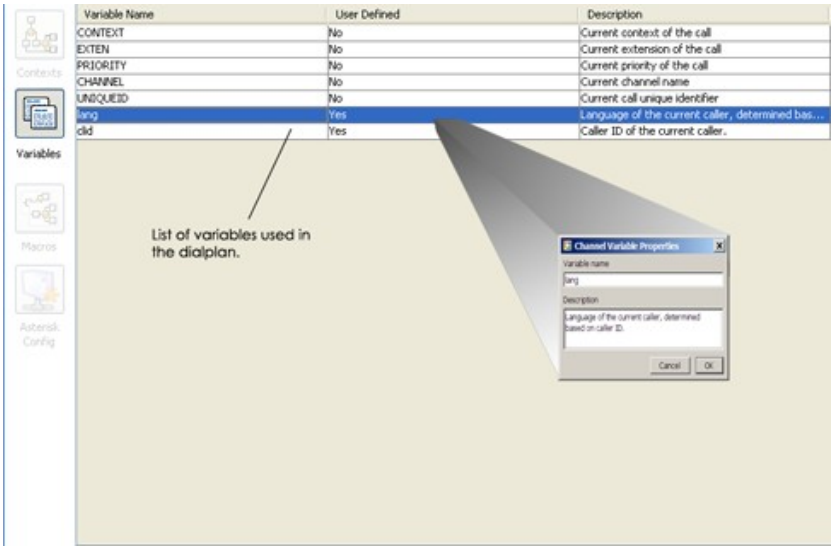


Figure 4 - Variables view

Macros/Subroutines view

The Macros/Subroutines button is the main access point to all the macros/subroutines used in the dialplan. Click on this button to switch the main window to the macros/subroutines view and to get a list of all macros/subroutines.

Any macros/subroutines that didn't pass the validation test will be shown in red, which means those macros/subroutines have some errors and should be fixed before deploying the dialplan.

By double clicking on the macro/subroutine name, selected macro/subroutine will be opened in the design panel.

Right click on the macro/subroutine name will open additional menu that will allow user to open macro/subroutine, create new macro/subroutine, or cut, copy, delete selected macro/subroutine, past copied macro/subroutine or edit macro/subroutine properties.

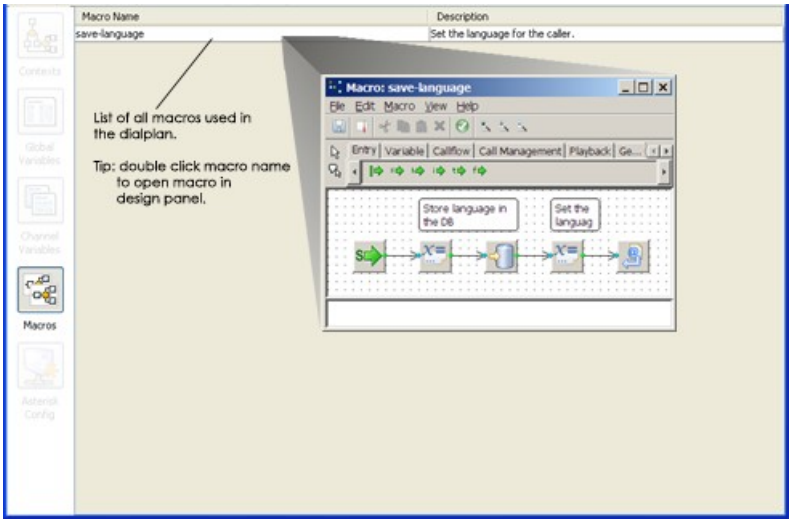


Figure 5 – Macros/subroutines view

Asterisk configuration view

This view displays the entire dialplan related configuration data collected from the Asterisk server.

By clicking on the left side list you can switch the view to present the list of SIP peers, IAX peers, Voicemails, Conferencing rooms, Music On Hold, Agents, Queues etc. This is a view only list, you can not modify configuration data through this view.

To enable Visual Dialplan to read configuration data you should define the connection to the Asterisk server. You can do that in the Preferences dialog. Note that connection to the Asterisk server is configured by default in Visual Dialplan PBX Edition.

Visual Dialplan will not modify the Asterisk configuration files, but will simply read the configuration data to simplify dialplan development.

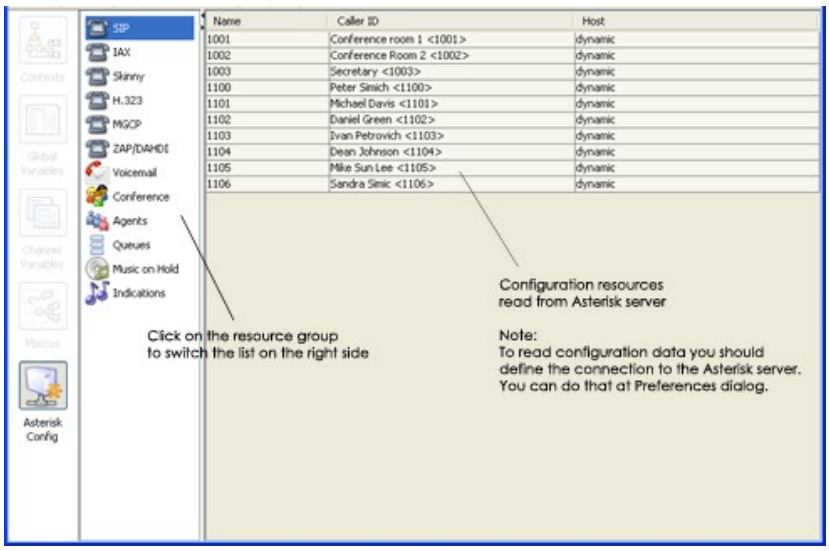


Figure 6 - Asterisk configuration view

Once you change the Asterisk server configuration simply click on the refresh button in the toolbar to read the latest server configuration and to make your dialplan aware of the latest configuration changes.

ASR Grammar View

All grammars used in the dialplan are listed under the ASR Grammars view.

Under this view you can create new grammars, edit existing grammars or delete grammars.
The LumenVox (ASR voice) related components (Load Grammar etc.) will be pre-populated with the grammars listed in this view.

To create a new grammar simply select *File/New* from the menu or click on the *New Item* button at the toolbar and new grammar will be created. The file name will be the same as the grammar name with a file extension of *.gram*. For example, if you created a grammar called *PhoneNumber* the file named *PhoneNumber.gram* will be deployed on the Asterisk server.

All grammars are saved together with the dialplan in the dial plan (VDP) file, not as separate files. Once you deploy the dialplan the grammars will be deployed too, this time as separate files (*PhoneNumber.gram*) and stored in the ***/etc/asterisk/grammars*** folder at Asterisk server.

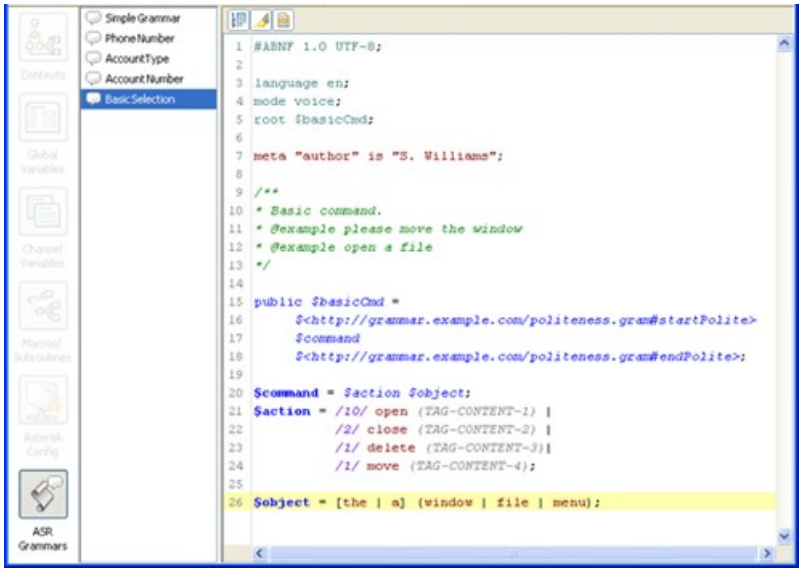


Figure 7 - ASR Grammar view

Integration Server View

Integration Server (IS) is powerful application server that extends Asterisk dial plan functionality by integrating it with third party servers and services (database servers, email servers, payment servers etc.). It comes with Visual Dialplan building blocks that provide an intuitive interface to easily access those third party servers and services, like execute SQL queries, send emails, process payment and more, directly from the dial plan.

Integration Server modules and resources (database connections, SQL queries, SMTP server connections, email templates etc.) are managed in this view. The building blocks located on the Integration Server sheet (IS) in Visual Dialplan will be pre-populated with the resources created on this view.

To create a connection to a new database server , email server or payment server simply right click the appropriate resource (*Database resource* or *Email resource or Payment resource*) and choose the New Item option from the drop down menu.

Integration Server resources (database connection, SQL queries etc.) are stored in the VDP file together with other call flow related information.

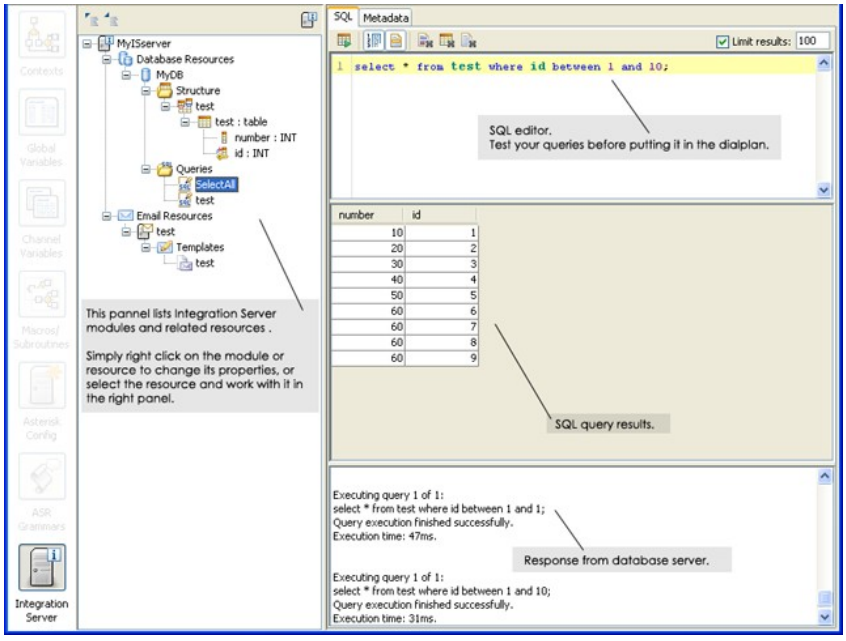


Figure 8 - Integration View

Grammar editor

The Grammar editor is integrated part of the ASR Grammar view. Simply select a grammar on the left side and the grammar editor will be opened on the right side of the window.

Grammar editor comes with syntax coloring and provides auto-complete suggestions. You can always get a drop down list of suggestions by pressing at *CRTL-Space*.

There are also the following options that may speed grammar development:

Show / hide line numbers.

Mark word occurrences across the document.

Highlight current line.

ASR Grammar Specification

The grammars are intended for use by speech recognizers such as (LumenVox Speech Engine, www.LumenVox.com) and other grammar processors so that developers can specify the words and patterns of words to be listened for by a speech recognizer.

You may refer to the W3C specification for complete details on writing grammars http://www.w3.org/TR/speech-grammar/, or to LumenVox grammar tutorial available at http://www.lumenvox.com/help/speechEngine/, but you may also find the ASR Grammar Specification document (part of this documentation) a useful tutorial in the general principles used in writing grammars.

Dialplan Includes

Dialplan include functionality allows user to include existing traditional Asterisk dialplan code or part of the existing traditional Asterisk dialplan code into Visual Dialplan and to keep maintaining that part of the dialplan in a traditional way by coding the dialplan outside of the Visual Dialplan.

Include functionality does not transform included Asterisk dialplan code into the Visual Dialplan i.e. it does not generate a graphical diagram based on the included dialplan code, but rather includes the dialplan code generated outside of the Visual Dialplan into the code generated with the Visual Dialplan.

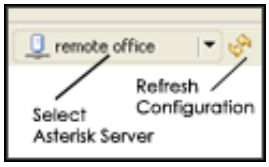
Important notice:

Include will happen at the Asterisk server side after deploying Visual Dialplan. In other words, the user is responsible for creating, maintaining and uploading traditional Asterisk dialplan code on the Asterisk box.

This functionality may significantly speed up the transition from traditional Asterisk dialplan coding to Visual Dialplan. You can use your existing dialplan code as an include, and port one by one your existing contexts into the Visual Dialplan.

Asterisk server selection

Under this drop down menu you can select the target Asterisk server you want to build the dialplan for and/or to deploy the built dialplan to. If you have configured several Asterisk servers, the five most recently used servers will be listed under this drop down box.



Note that in order to take full advantage of this product you must select a target Asterisk server and that way enable Visual Dialplan to read the Asterisk server configuration and accommodate its behavior accordingly.

If you changed the configuration of your target Asterisk server manually or using any other GUI tool, please be sure to let Visual Dialplan know about the configuration change by clicking on the "Refresh configuration" button.

Contexts and Macros editor

The Contexts and Macros editor is a graphical view used for context and macro visual modeling. Here you can build and edit a context or macro by placing and connecting dialplan building objects, and defining their properties. The Design panel can be accessed from the Context or Macros view from the Main window by double-clicking the context or macros name.

This panel consists of the menu, toolbar, component toolbar with dialplan building objects, visual modeling area and info area.

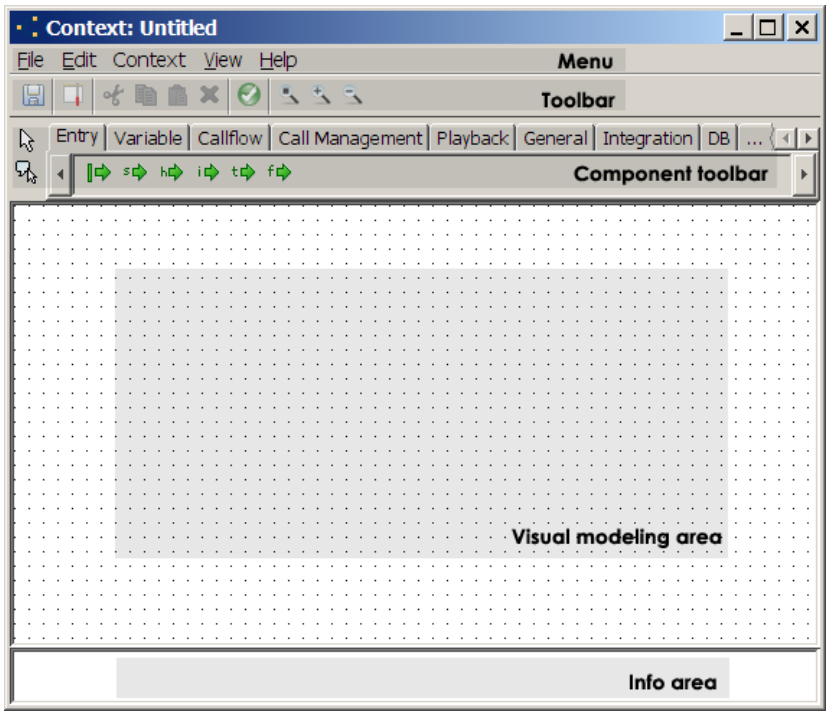


Figure 9 - Design panel organization

Menu

The following menus are available on the design panel:

File menu

- Save (ALT-F-S)
  Save currently opened context (macro) and continue work on it.
- Save and Close (ALT-F-A)
  Save currently opened context (macro) and close the design panel window.
- Properties (ALT-F-P)
  Edit context (macro) properties, basically name and description.
- Export (ALT-F-E)

  Export the context or macro graphical representation to the image file.
- Close (ALT-F-C)
  Close design panel window. In case you modified the context (macro) and didn't save it before closing, application will offer you the option to save context (macro) before closing it.

Edit menu

- Undo

Undo mistake (return one step back).

- Redo

Redo mistake (go one step forward).

- Cut (CTRL-X)
  Removes selected objects and moves the objects to the clipboard.
- Copy (CTRL-C)
  Copies selected objects to the clipboard.
- Paste (CTRL-P)
  Moves copied objects from the clipboard and create the same objects.
- Delete
  Deletes selected objects.

Context (Macros) menu

- Include
  This option allows you to include another context into the current context or macro.
- Validate
  This option will check currently opened context (macro). Components that are not properly connected or configured will be marked in red and more descriptive error message will be placed in the Info area.

View menu

- Actual Size
  Set graphical presentation to the default size for the dialplan building objects and the working grid.
- Zoom in
  Zoom in graphical presentation.
- Zoom out
  Zoom out graphical presentation.

Help menu

- Help Contents (F1)
  Invoke content sensitive help.

  Simply select (click on) the object you want to get the help for and click the F1 on the keyboard.

Toolbar

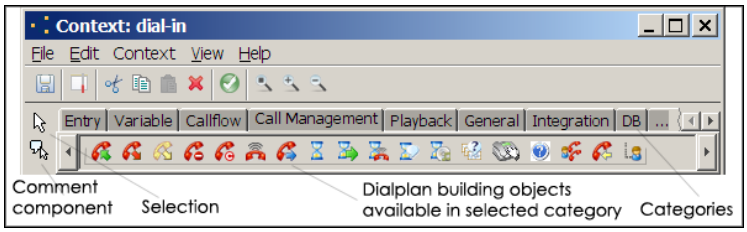
The following toolbar icons are available on the design panel:

- Save
  Save currently opened context (macro) and continue work on it.
- Properties
  Edit context (macro) properties, basically name and description.
- Undo

  Undo mistake (return one step back).
- Redo

  Redo mistake (move one step forward).
- Cut
  Removes selected objects and moves the objects to the clipboard.
- Copy
  Copies selected objects to the clipboard.
- Paste
  Moves copied objects from the clipboard and create the same objects.
- Delete
  Deletes selected objects.
- Include
  This option allows you to include rules of another context into the current context or macro.
- Validate
  Check currently opened context (macro) against component connections or configuration errors.
- Actual Size
  Set graphical presentation to the default size.
- Zoom in
  Zoom in graphical presentation.
- Zoom out
  Zoom out graphical presentation.

Component toolbar



The Component toolbar is a place holder for all the dialplan building objects – dialplan components (Asterisk applications) and functions.

Building objects are organized into the categories, visible as sheets at the component toolbar.

Each building object is presented with an icon. To get a name of the building block just position the cursor over the icon, or place the building block on to the visual modeling area and press F1 to get detailed help about that particular component or function.

There are two special components that are used for selections and comments:

Selection

Comment component used for placing comments on the visual modeling area

Visual modeling area

This is the area (grid) where all the visual modeling happens. You can place as many dialplan objects as you wish, connect, configure and organize them into contexts and macros to model any kind of call flow.

See 'Placing, connecting and setting up building objects' section of this manual for more details.

Visual modeling area is significantly larger than it looks like when you open it first time. You can use Zoom In, Zoom Out and Actual Size icons on the toolbar to reach any part of the visual modeling area, and to create contexts and dialplans significantly larger than the initially visible area size.

While working with building blocks in the visual modeling area you can select, cut, copy, paste and delete group of building objects and move them between different contexts and macros.

Info area

The Info area appears across the bottom of the design panel window and displays information about highlighted building block in the Info sheet.

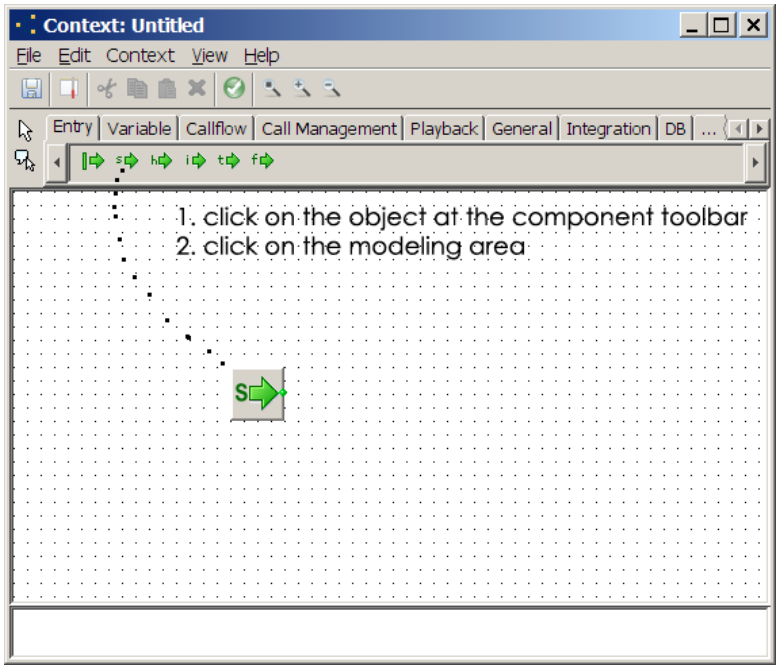
In case the building block is not connected or configured properly, error or warning info will be presented in the Problems sheet in red once the building block is highlighted.

In the Build sheet you will also get the native Asterisk dialplan code generated in the background while you visually model or edit building block.
Simply open Build sheet and watch how Visual Dialplan generates Asterisk dialplan code based on the selected component and its settings.

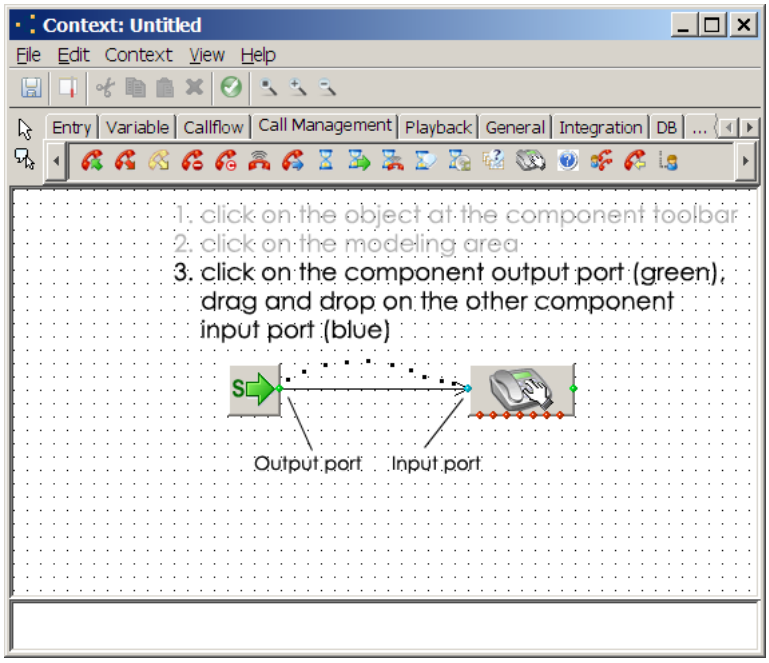
Placing, connecting and setting up building objects

The Dialplan building objects are located on the component toolbar. Simply click on the building block on the component toolbar to select the component and then click on the visual modeling area (working grid) to position the component on the modeling area. Then you can connect the component to another one by clicking on the component output port (green port) and dragging/dropping connection (line) to other component input port (blue port), or the other way around. You can also double click on the component located at the modeling area to get the component property editor (if available for that particular component) and set the component behavior.

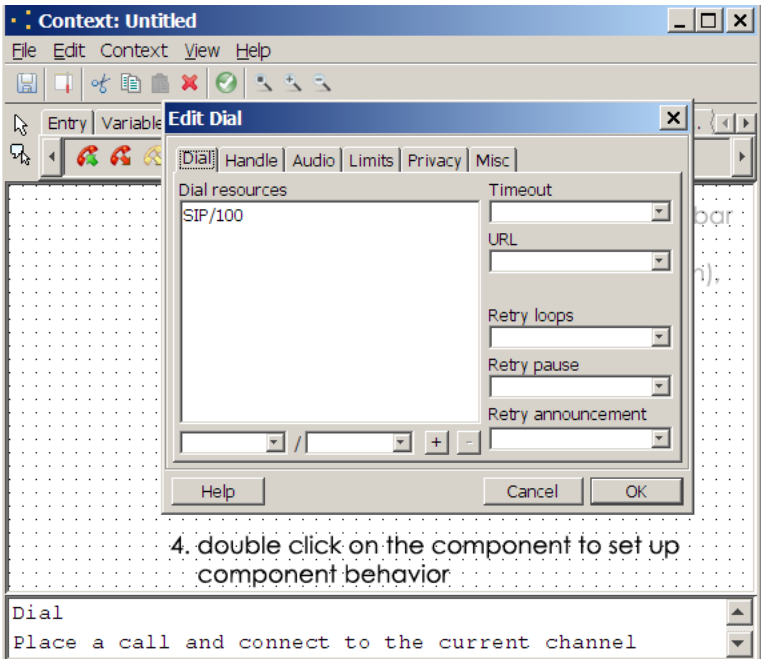
Step 1 – place building object on the modeling area



Step 2 – connect one component to another



Step 3 – Set component behavior



This way you can place every dialplan building object on the visual modeling area, connect those blocks and set up each component behavior.

In addition, if you are not sure about some component behavior, simply select that component (single click on the component located at the visual modeling area) and press F1 on your keyboard to get detailed help about the component.

Inserting comments

The Comment component allows the user to insert notes about part of the dialplan or some particular object within the dialplan. Comments can be inserted anywhere in the visual modeling area.

Comments appear as text boxes. A user can move or edit a comment at any time.

To insert a comment:

1. Right-click on the Comment component
2. Position the pointer where you want it, and then right-click. A new comment box appears.
3. Double-click the comment box and then type in your comment.
4. When you are finished, click anywhere outside the comment box or press Enter.

You can now move the comment box by dragging it, or resize it to fit your needs.

Properties editor

The behavior of each Visual Dialplan component or function can be fine tuned by setting its properties.

Visual Dialplan provides specialized editors for each component. Just double click the component from the working area and a property editor will appear. You don't have to know the syntax and properties for each and every dialplan application. Property editors provide an easy and intuitive way for setting these values.

Expression editor

The Expression editor is a special version of the property editor, commonly used in functions, designed for setting up complex expressions and assigning values (see figure 8).

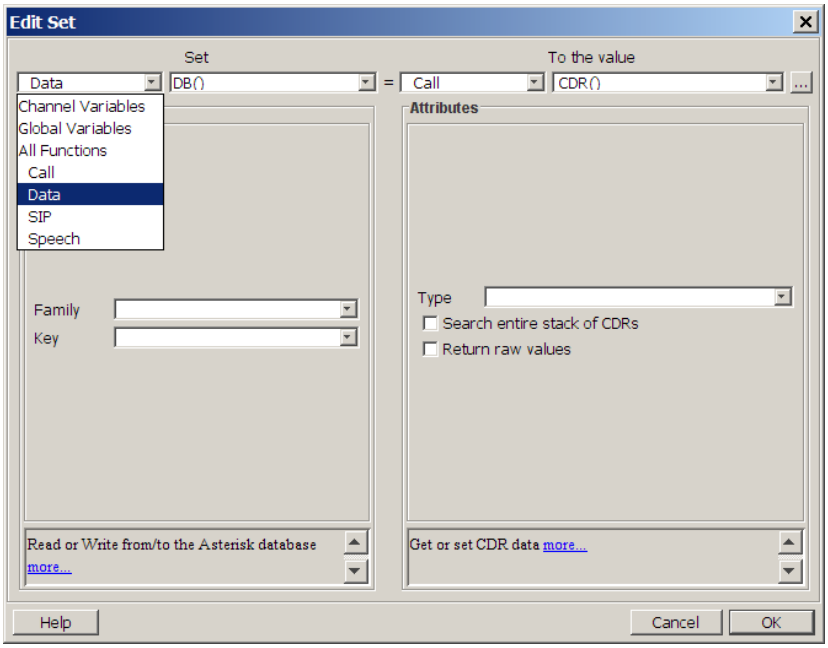


Figure 10 - Expression editor

Typical steps to assign a value (variable or function result) are:

1. Select a group of variables or group of functions from the Set drop down box (first drop down from the left)
2. Select the exact variable or function that will store the expression result value (second drop down from the left), or select drop down and click on Ctrl-n to create new variable on the fly.
3. The Function property editor will be opened below the drop down menu waiting for you to select/enter property values (DB() function property editor is opened in sample presented above).

   Basic info about the selected function is displayed at

the bottom of the window. You can always click on the link '*more…*' to get detailed info about the selected function.

4. Now you should repeat the same steps for the expression (drop downs on the right side of the window). Fist select variables or functions group and then exact variable or function. And last but not least, fill in the function property values in case you selected an function.

   Basic info about the selected function is displayed at the bottom of the window. To get detailed info about the function click on the '*more…*' link.

This approach could be used to assign a single value (variable or function result), but if you want to execute complex expressions and assign the expression result, you will need to use the advanced expression editor accessible at  button presented on the right (see figure 8).

The Advanced expression editor is very similar to the standard expression editor with a larger input area at the top of the windows.

Expression could be created using drop down menus and function property editors, pretty much the same like in the standard expression editor, or by typing expression directly in the expression input area.

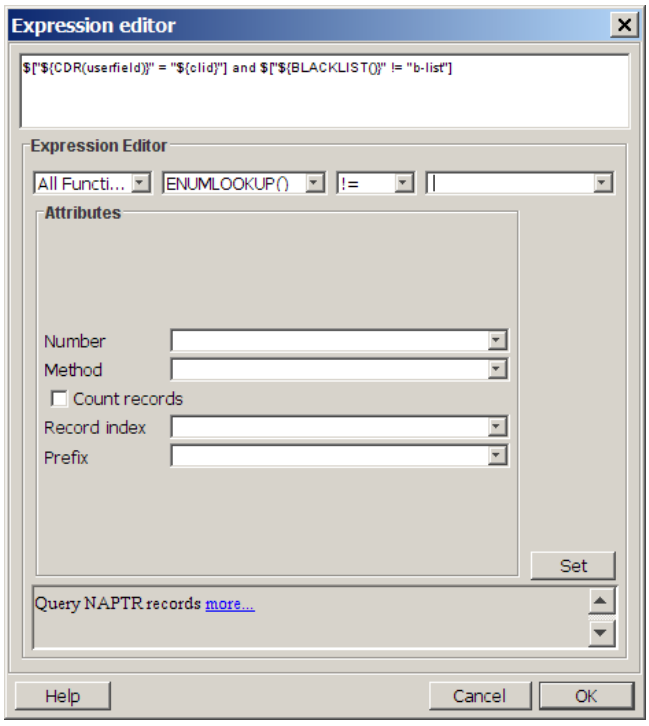
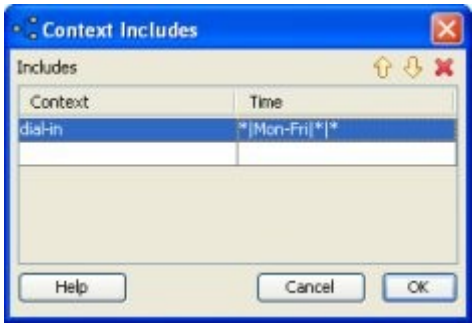


Figure 11 - Advanced expression editor

Basic info about currently used function is displayed at the bottom of the window. You can always click on the link '*more…*' to get detailed info about the selected function.

Context/Macro Includes

Context/Macro include functionality allows you to include rules of another context into the current context or macro.



In addition, you can define the time frame, days of the week, days of the month and months when the included rules should take place.

Export to image file

To export the call flow graphical representation, select the Export option under the File menu in the Contexts or Macros editor window.
You can export the graphical representation as an image file or SVG file.

Deploying the Dialplan

Simply select 'Dialplan' then 'Deploy' option from the menu or click on the 'Deploy' button at the toolbar and Visual Dialplan will generate the standard Asterisk dial plan code (extensions.conf code) based on the currently opened visual dialplan and will deploy it to the server.

Preferences

The Preferences dialog allows user to configure different aspects of the Visual Dialplan application. Using the preferences dialog user can configure the following application parameters:

- Asterisk server connection preferences
- Dialplan preferences
- General Visual Dialplan parameters

Asterisk server connection preferences

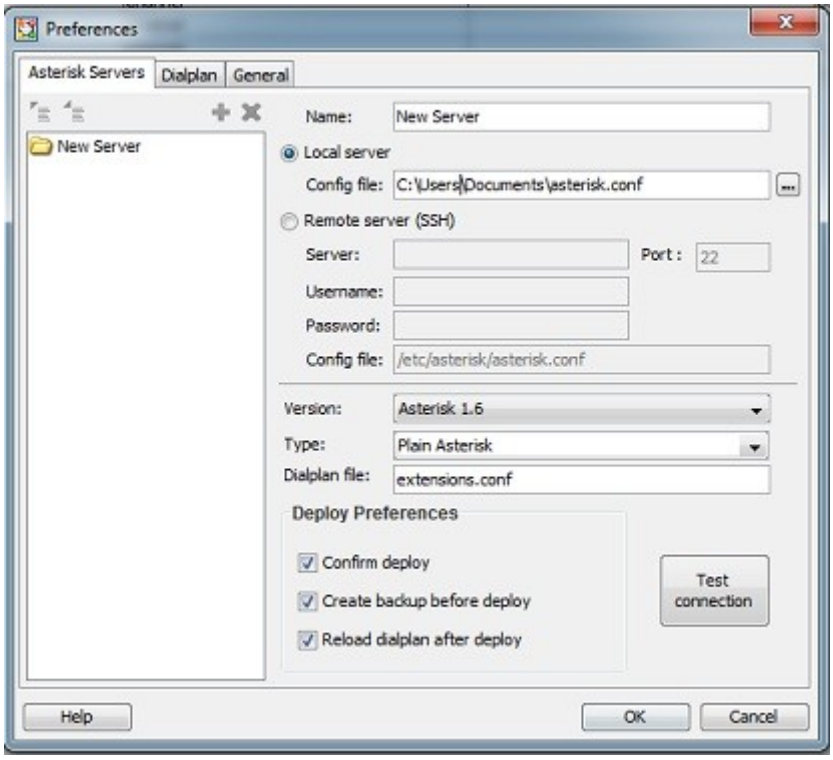
This dialog enables user to define and manage connections to the target Asterisk server(s).

Note that in Visual Dialplan PBX Edition the connection to the PBX server is configured by default and cannot be changed, nor additional connection to additional PBX servers can be made.

Visual Dialplan Professional support a plain Asterisk server (manually configured server without GUI) as well as all major Asterisk based GUI distributions, like Trixbox, PBX In a Flash and Elastix among others.

You can choose between the Local and Remote Asterisk server connection. Use the Local option if you are using the Visual Dialplan Professional on the same box where your Asterisk server is deployed. Use the Remote option if you are developing the dialplan to a remote Asterisk server.

The following figure and table describes all properties of the Asterisk server preferences.



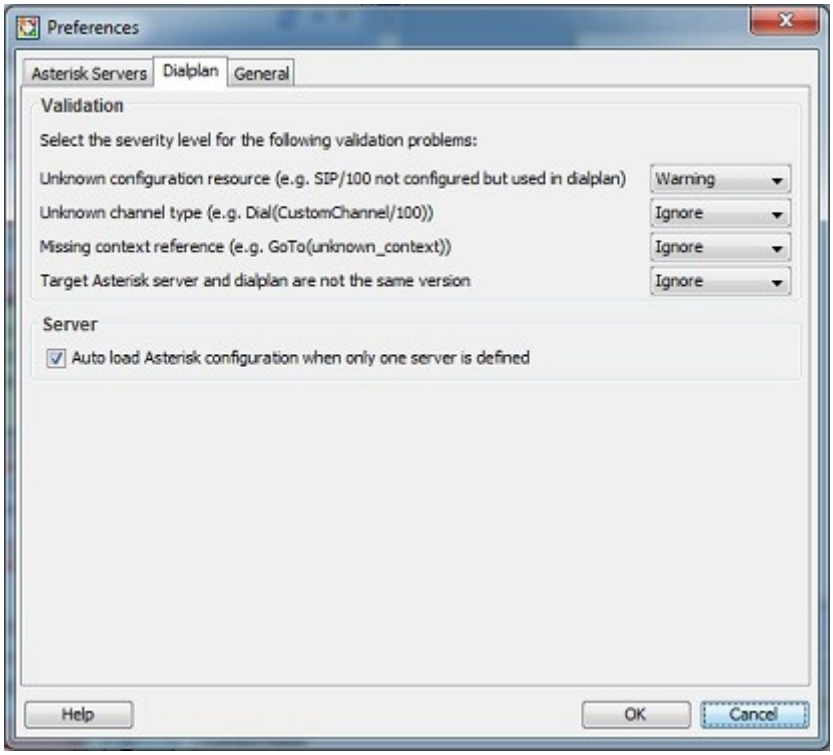
Field	Description
Name	Descriptive name of target server.
Version	Click on the "Test connection & Detect version" button to instruct Visual Dialplan to try to read the Asterisk server version number by connecting to the Asterisk server.
Local server and Config file	Select this option if your Asterisk server is deployed at the same box where the Visual Dialplan is deployed. And enter the path to the Asterisk.config file under the "Config file" field.
Remote server (SSH)	Select this option if you are developing dialplan for a remote Asterisk server.
Server and port	URL or IP of remote Asterisk server and port to connect to with integrated SSH client (default is 22).
Username	The username that integrated SSH client will use to connect to the remote Asterisk box.
Password	The password that integrated SSH client will use to connect to the remote Asterisk box.
Config file	Path to the asterisk.config file.
Asterisk server type	Visual Dialplan supports vanilla Asterisk servers (manually configured server without GUI) and all major Asterisk based GUI distributions, like Trixbox, PBX In a Flash and Elastix among others. Select the Asterisk server type or click on the "Test connection" button to instruct Visual Dialplan to try to read the Asterisk server

	type by connecting to the Asterisk server.
Confirm deploy	If checked this field will instruct Visual Dialplan to ask you to confirm dialplan deployment each time you click on the deploy button. We strongly recommend you to have this button checked.
Create backup before deploy	If checked this field will instruct Visual Dialplan to create a backup of extensions.conf file before deploying new (overwriting the existing) extensions.conf file. We strongly recommend you to have this button checked.
Reload dialplan after deploy	If checked this field will instruct Asterisk server to reload extensions.conf file after the dialplan deployment. This way the dialplan changes will immediately take effect.
Dialplan file	Output of the Visual Dialplan is standard Asterisk extensions.conf file. This field should contain the path and the name of the target Asterisk extensions.conf file.

Button	**Description**
Add new server	Define new Asterisk server.
Remove server	Delete already defined Asterisk server and release the license associated with that Asterisk server.
Export server definitions	Export defined server definitions.
Import server definitions	Import server definitions.

Dialplan Preferences

Using this dialog you can define various dialplan development related options.
Under the "Validation" group you can fine tune behavior of the Validation engine and define how Visual Dialplan should report various validation errors and warnings.

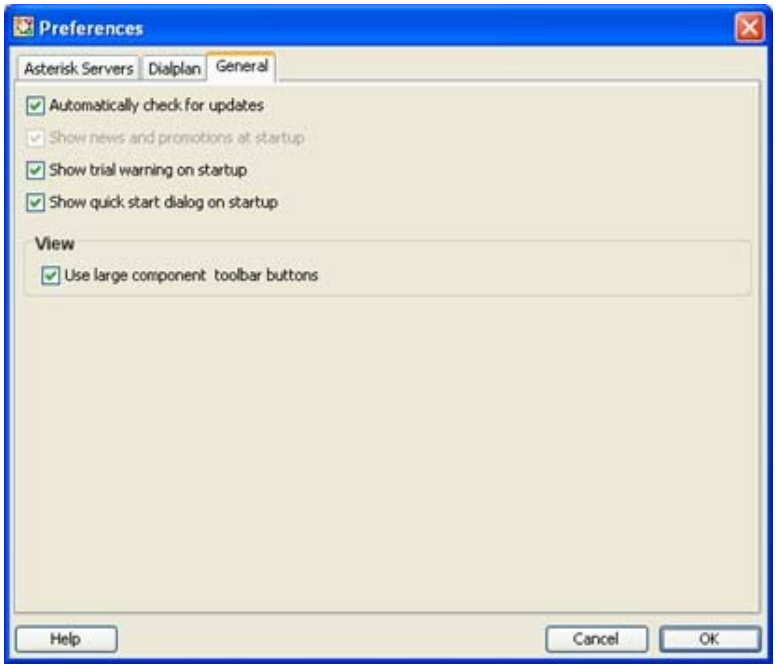


Field	Description
Unknown configuration resource	Under this field you can define how Visual Dialplan should report missing configuration resources. For example, if you use the SIP/100 resource in the dialplan but you do not have a SIP/100 resource defined in the Asterisk server configuration, Visual Dialplan will recognize it and report it based on the settings in this field.
Unknown channel type	Under this field you define how Visual Dialplan should report missing channel type. For example, if you use GTALK/100 resource in your dialplan but the GTALK channel type is not recognized by the Asterisk server, Visual Dialplan will report it based on the settings in this field.
Missing context reference	This field defines how Visual Dialplan should act in case of the missing dialplan context reference. For example, if you reference missing context (Goto(missing-context)), Visual Dialplan will report it based on the settings in this field.
Target Asterisk server and dialplan are not the same version	Instruct Visual Dialplan how to report difference in versions between the dialplan and the target Asterisk server. Visual Dialplan may report only difference between the main versions.

Auto load Asterisk configuration when only one server is defined	This option instructs Visual Dialplan to load Asterisk server configuration automatically when there is only one Asterisk server connection defined.

General Preferences

This dialog enables you to set general preferences like automatic check for updates, behavior of Quick start window and similar.



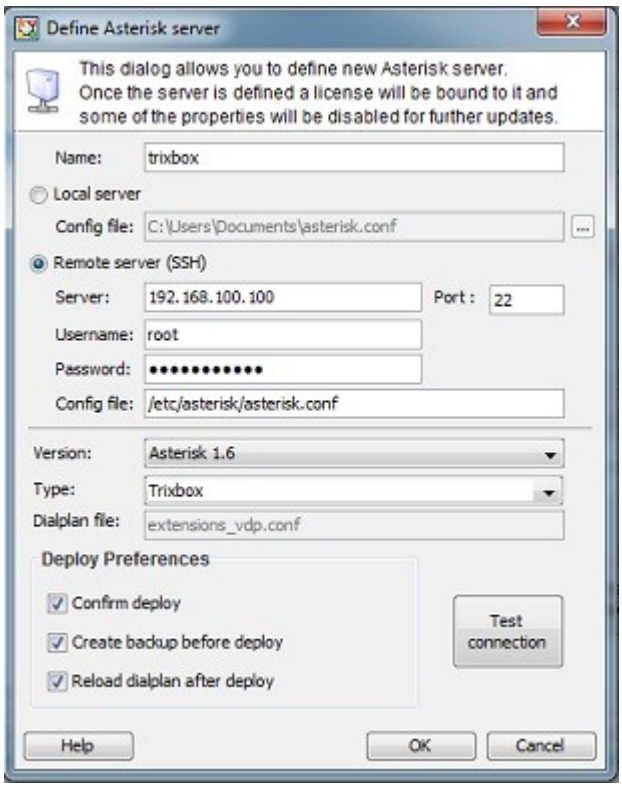
Field	Description
Automatically check for updates	If checked this field will instruct Visual Dialplan to check for updates each time the application is started.
Show news and promotions at startup	If checked this filed will instruct Visual Dialplan to present targeted news and promotions, if available, at application startup.
Show trial warning on startup	If checked you will get the trial warning windows at application startup, unless the application is registered. Deselect this value if you want to automatically skip that info at application startup.
Show quick start dialog on startup	If checked this field will instruct Visual Dialplan to present you the Quick start window at application startup. Quick start window offers you to start new dialplan, open existing dialplan or to open some of provided sample dialplans from the included library.
Use large component toolbar buttons	If checked this field will instruct Visual Dialplan to present the large component toolbar buttons with component name

	below the button instead of small toolbar buttons. View with large component toolbar buttons is recommended for new users.

Define New Asterisk Server

Note that this functionality is not available in Visual Dialplan PBX Edition. In PBX Edition Visual Dialplan comes with predefined connection to the PBX server and that connection cannot be modified nor additional connections to additional PBX servers can be made.

Using this dialog you can define a new Asterisk server. This dialog can be accessed by selecting Edit/Preferences... option from the main menu. Once the Preferences dialog appears click on the plus button at the top of the list to define new server.

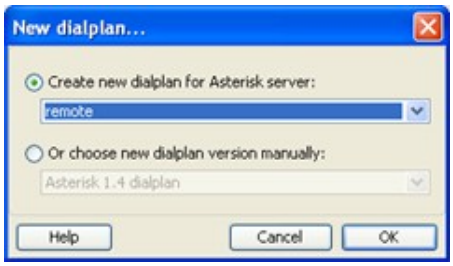


Create new dialplan

To create a new dialplan select 'File > New dialplan' from the main Visual Dialplan window.

You can create a new dialplan by selecting the target Asterisk server from the list of available Asterisk servers, or by selecting Asterisk dialplan version from the list of supported dialplan versions.

Our recommendation is to start a new dialplan by selecting the target Asterisk server and that way enable Visual Dialplan to read your Asterisk servers configuration and accommodate its behavior accordingly.



Upgrade/downgrade the dialplan

To upgrade or downgrade the dialplan select 'Dialplan > Version...' from the main Visual Dialplan window. You will be presented with the dialplan version number and in the drop down menu you will see a list of supported Asterisk dialplan versions. To upgrade or downgrade the currently open dialplan simply select the dialplan version you want to upgrade/downgrade to and click on the 'Upgrade' button. Although we try to automate this procedure as much as possible, it can not be completely automated and after the upgrade/downgrade you will probably need to update some of the dialplan components manually. It is highly recommended to make a backup copy before upgrading/downgrading the dialplan because there is no simple way to revert back to the previous state after this action.

Chapter

4

Visual Dialplan Components

Visual Dialplan components are organized on the component toolbar in the following categories:

- Entry
- Variable
- Call flow
- Call management
- Playback
- Integration Server
- General
- Exe
- Ast Addons
- VM & Conf
- Queue
- Recording
- Caller ID
- Billing
- LumenVox
- Zap/Dahdi
- Misc

In this chapter we will describe in more details each component.

Entry category

Entry category contains extensions related components.

Extension

Defines custom extension for this context

This is a commonly used component to define custom extensions for the context.

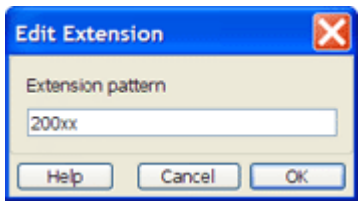
There are several special characters that may be used to create extension pattern:

- X - matches any digit from 0-9
- Z - matches any digit from 1-9
- N - matches any digit from 2-9
- [1237-9] - matches any digit or letter in the brackets (in this example, 1,2,3,7,8,9)
- . - wildcard, matches one or more characters
- ! - wildcard, matches zero or more characters immediately

For example, extension patterns could be:

- 200 to define extension number 200
- vip to define extension vip
- _200xx to define any extension starting with 200 and following with any other two digits
- _3. to define any extension that starts with digit 3 and continues with any other digits

Property Editor



Field	Description
Extension pattern	A custom defined extension pattern.

StartExten

Defines the start extension for this context

Start extension is primarily used for dialplans that enter a context without extension information. Think of it as a no DID phone line, and when call comes in we may only know that the line is ringing, nothing else. Start extension is also usually used when the caller have to pass through some common part of the dialplan before proceeding to particular extension.

HangupExten

Defines the hangup extension for this context

This extension is reached when caller hangs-up a call.
The extension could be used to clean up a call or to play a goodbye message before hanging up. This component is also used by the calling card software to signal the end of the call for billing purposes.
Please note that this component will not work if the call is parked.

InvalidExten

Defines the invalid extension for this context

This extension is reached when caller dials an unknown extension in a context or enters unknown input at an IVR menu.

TimeoutExten

Defines the timeout extension for this context

This extension is reached when caller has been inactive after a prompt was played i.e. didn't type in extension or didn't select IVR option after predefined time (timeout time). The timeout extension is also used to hang up a line that has been idle for some period of time.

FaxExten

Defines the fax extension for this context

This extension is used for fax detection on Zap channels.

Variable category

The Variable category contains components for setting up dialplan variables and Asterisk functions.

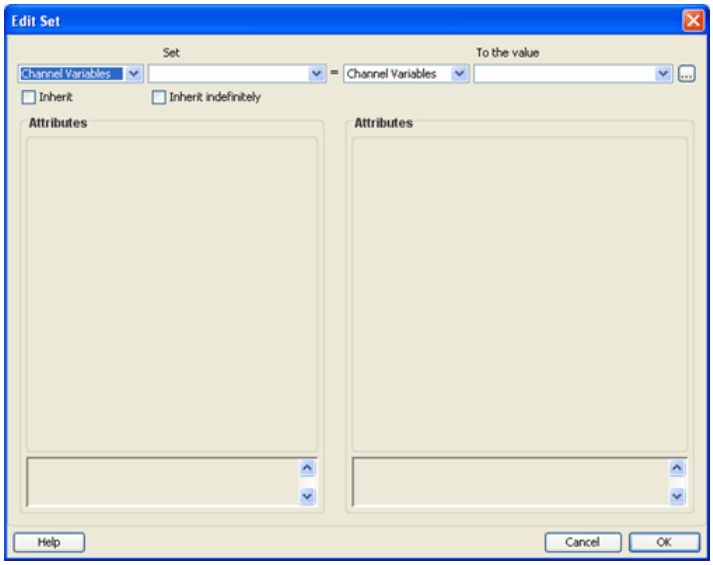
Set

Set or get channel variable or function value (over 40 functions)

This component is used to set or get the value of channel variables or dialplan functions. There are more than 40 functions that can be accessed using this component.

Note: By selecting 'Set' drop down and clicing on Ctrl-n you can create new channel or global variable on the fly.

Property Editor



Field	Description
Set	Variable to set. You can select between: Channel variables Global variables Functions (over 40 functions)
To the value	The value to set the variable to. You can select between: Channel variables, Global variables or Functions.
Inherit/Inherit indefinitely	If the variable name is prefixed with _, the variable will be inherited into channels created from the current channel. If the variable name is prefixed with __, the variable will be inherited into channels created from the current channel and all children channels.

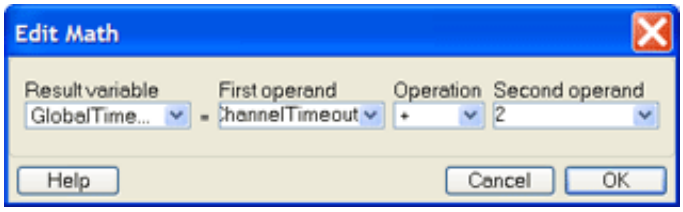
Math

Performs mathematical functions.

This component will execute mathematical operations: sum, multiply, divide, substract, modulus, grather then, less then, greather then or equal, less then or equal, equal.

Note: By selecting 'Result variable' drop down and clicing on Ctrl-n you can create new variable on the fly.

Property Editor



Field	Description
Result variable	The variable that will store the result.
First operand	First operand.
Operation	Math operation.
Second operand	Second operand.

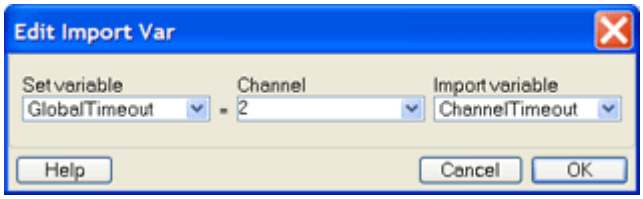
Import Var

Import a variable from another channel (not from the current channel)

This component imports a variable from the specified channel and stores it as a variable in the current channel. Variables created by this component have the same inheritance properties as those created with the Set component.

Note: By selecting one of the drop downs and clicing on the Ctrl-n you can create new variable on the fly.

Property Editor



Field	Description
Set variable	Set Asterisk variable.
Channel	Choose channel.
Import variable	Import variable from selected channel.

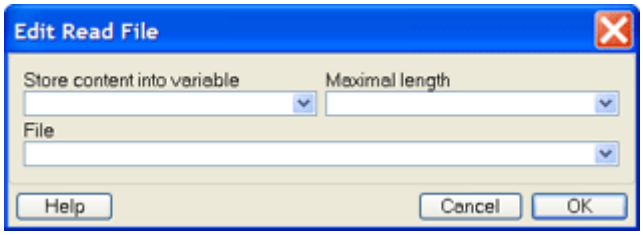
Read File

Read the content of a file and load into a variable

This component reads the content of a file and stores it in a variable.

Note: By selecting one of the drop downs and clicing on the Ctrl-n you can create new variable on the fly.

Property Editor



Field	Description
Store content into variable	Set Asterisk variable.
Maximal length	Maximum length of the variable.
File	A file to import variable from.

Call flow category

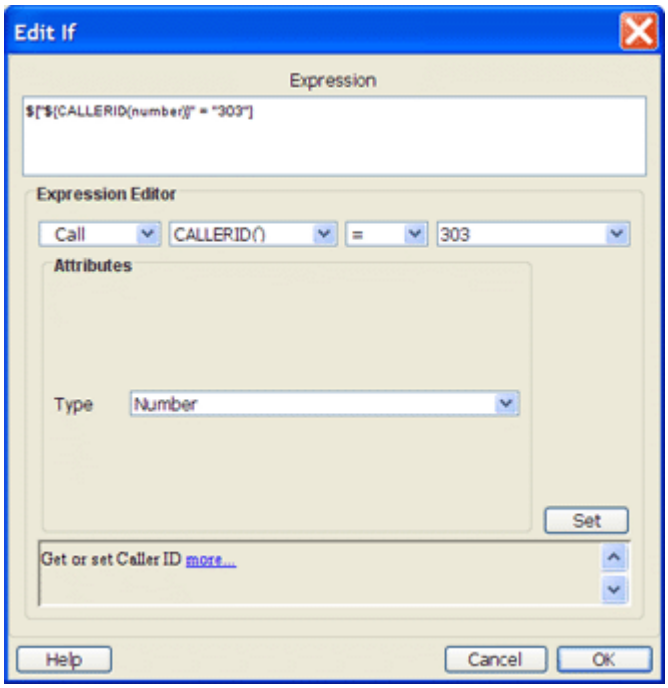
This category contains components for creating branches in the call flow and calling macros and/or subroutines.

GotoIf

Conditional branching of call flow

This component will branch the call flow based on the expression result.

Property Editor



Field	Description
Expression	The expression is a string. If the string is empty or "0", the condition is considered to be false. If it is anything else, the condition is true.

Output options

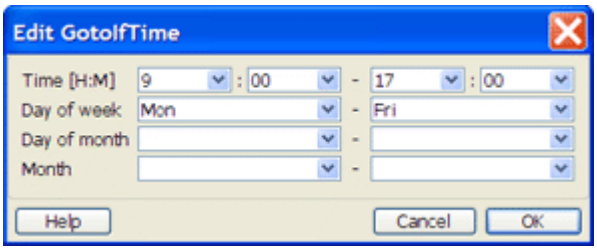
Output	Description
Continue	The condition is considered to be false.
Branch	The condition is considered to be true.

GotoIfTime

Branch based on current time

This component will branch the call flow based on the current time. If the current time matches the specified value, the call flow will branch, otherwise the call flow will just continue on the next step. Each of the elements may be specified either as '*' (for always) or as a range.

Property Editor



Field	Description
Time	Time in format HH:SS (hours:seconds).
Day of week	Day of week (Sunday through Saturday).
Day of month	Day of month (1 through 31).
Month	Month (January through December).

Output Options

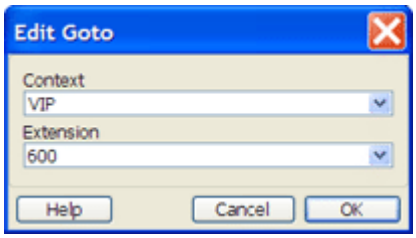
Output	Description
Continue	The condition is considered to be false.
Branch	The condition is considered to be true.

Goto

Jump to a particular extension or context

This component will stop the current call flow and will continue call flow from a particular extension in a selected context (jump to a particular extension or context).

Property Editor



Field	Description
Context	A context to jump to.
Extension	Extension in previously selected context where the call flow will continue with the execution (jump to).

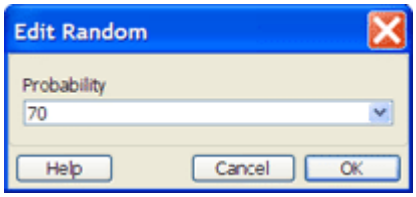
Random

Conditionally branch base on probability

The component will branch the call flow based on probability : 1 to 100(similar to percentage: 1% to 100%). A probability of 1 will take the branch about 1% of the time; 5, 5%, 10, 10%, etc. A probability of 100(or above). The component is located on the Callflow shee.

Note: This component could be used in load balancing purposes, for example to load balance inbound calls against different queues.

Property Editor



Field	Description
Probability	Branch the call based on this probability. A probability of 1 will take the branch about 1% of the time: 10, 10%, etc. A probability of 100 (or above) will always branch and is equivalent to Goto component.

Output Options

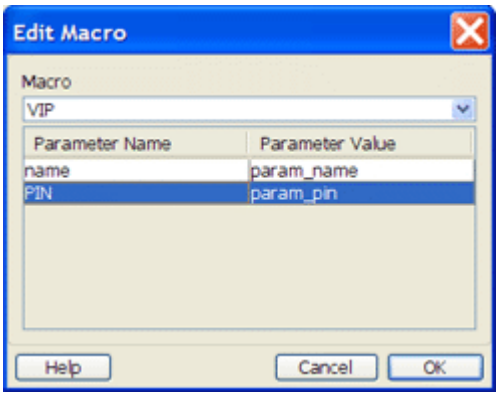
Output	Description
Success	The call flow continued in the main path (usualy a result of the low probability).
Failed	The call flow continued in the main path(usually a result of the high probability).

Macro

Execute a macro

This component will execute a macro defined under the Macros in the Dialplan view, executing each step, then returning when the steps end.

Property Editor



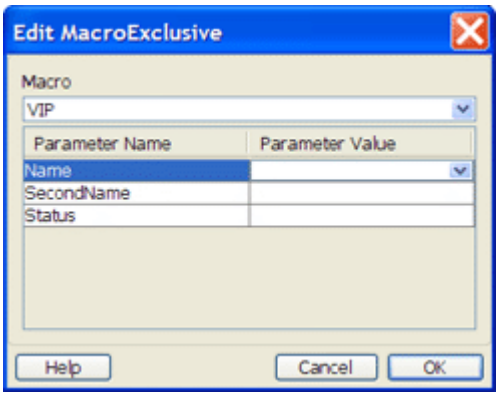
Field	Description
Macro	The macro as defined under the Macros in the Dialplan view.
Parameter name/value	Expected arguments of the macro in name/value format.

MacroExclusive

Exclusive Macro Implementation

This component executes macro as defined in the context 'macro-macroname'. Only one call at a time may run the macro, if the macro is executed using this component. The new call will wait if some previous call is executing the Macro.

Property Editor



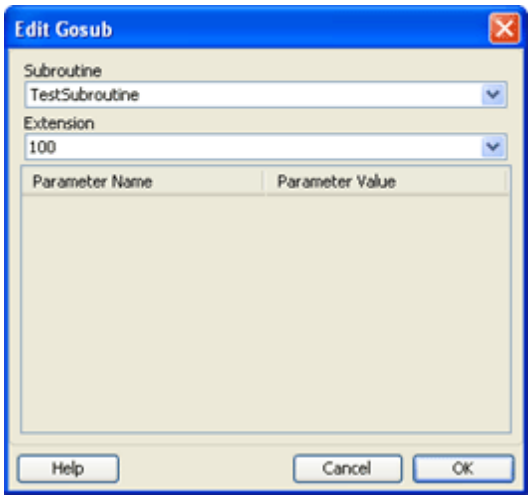
Field	Description
Macro	Select a macro that will be exclusively executed.
Parameter Name, Parameter Value	Macro parameters listed as a key-value list.

Gosub

Jump to subroutine

This component is used to jump to a subroutine from the call flow. The Return component is used to return back from the subroutine.

Property Editor



Field	Description
Subroutine	A subroutine where the call flow will jump to.
Extension	Extension of the subroutine in previously selected context.
Parameter Name and Value	Parameters in form of key-value pairs.

Return

Return from a subroutine

This component is used to return control back from the subroutine to the main call flow. The component goes together with Gosub component (used to jump to the subroutine).

StackPop

Remove a return address from the subroutine without returning

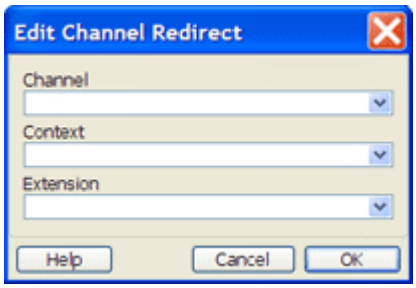
This component removes a single address from the stack of addresses accumulated by a series of Gosub components, but does not return to that address.

ChannelRedirect

Redirects given channel to a dialplan target

This component can be used to send a specified channel to the specified extension in the dialplan. It will also set the CHANNELREDIRECT_STATUS variable to SUCCESS upon success or NOCHANNEL in case the given channel does not exist.

Property Editor



Field	Description
Channel	A channel that will be redirected.
Context	Target context.
Extension	Target extension.

Switch PBX

Forward to another server

This component can be used to forward calls to another server.

Call management category

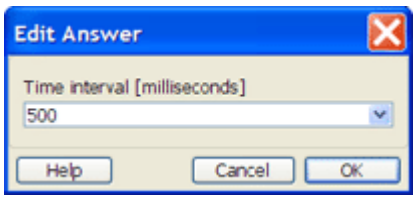
The Call management category contains components responsible for managing established calls at the channel like answer, dial, hang up, wait, music on hold etc.

Answer

Answer a channel if ringing

This component will answer a ringing channel. If a time interval is specified, Asterisk will wait for a specified number of seconds before answering a channel.

Property Editor



Field	Description
Time interval	Wait for specified number of milliseconds before answering the call.

Hangup

Unconditionaly hangup.

Unconditionaly hangs up a given channel.

Dial

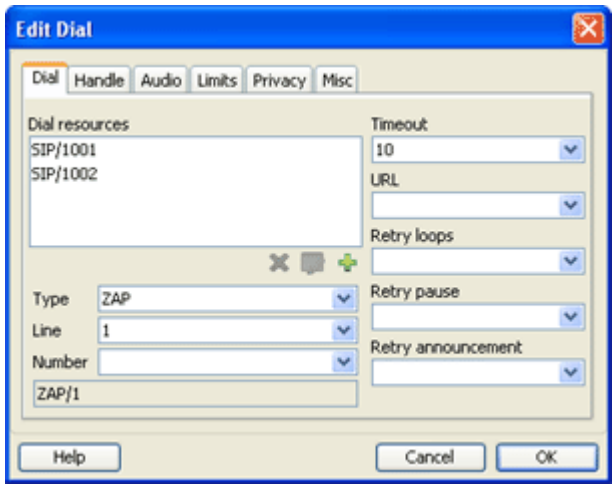
Place a call and connect to the current channel

This component attempts to establish a new outgoing connection on a channel, and then link it to the existing input channel. The originating channel that triggered this component is then answered, and the two channels are connected together ("bridged") allowing a conversation to take place. When the channel that triggered this component hangs up, the Dial component exits.

Output Options

Output	Description
Success	Dial statement was successful.
Unavailable	There is no one logged in on the called extension.
Congestion	Congestion on the channel.
No answer	The called party didn't answer (available if screening mode is enabled).
Busy	The dialed extension is busy.
Cancel	The call is canceled.
Don't call	Don't call (available if screening mode is enabled).
Torture	Torture (available if screening mode is enabled).

Property Editor - Dial

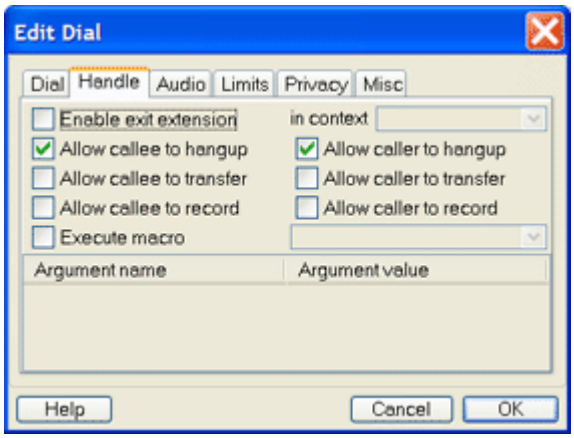


Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

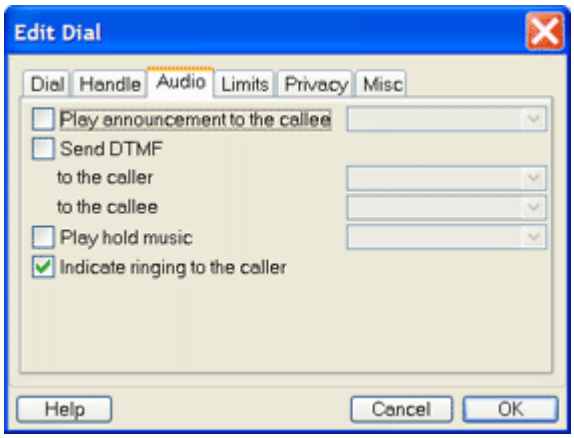
Field	Description
Dial Resources	A list of phones/end_points to be dialed given in format technology/resource.
Timeout	This is an optional parameter. If not specified, the Dial component will wait indefinitely, exiting only when the originating channel hangs up, or all the dialed channels return a busy or error condition. Otherwise it specifies a maximum time, in seconds, that the Dial command should wait for a channel to answer.
URL	This is an optional parameter and defines the URL to be sent to the called party once the line is successfully established. This option will work only if the channel supports sending URLs.
Retry loops	A number of times to try to place the call in case the channel can't be reached. In case the Retry loops is set to '-1', the call will retry endlessly.
Retry pause	The pause between two sequent tries to place the call in case the channel can't be reached. In case the value is set to less than 1 sec., pause will be set to 10 seconds.
Retry announcement	The announcement to be played each time the Dial component tries to place the call (as defined in the Retry loops option).

Property Editor - Handle



Field	Description
Enable exit extension	This option will enable reading caller entered digits while waiting for the call to be answered and will process that value further. This option is usually used to dial 1-digit exit extension while waiting for the call to be answered.
Allow callee to hangup	Allow called party to hang up by dialing *.
Allow callee to transfer	Allow called party to transfer the call by hitting the blind xfer keys (features.conf).
Allow callee to record	Allow called party to start recording after pressing *1 or as defined in the features.conf.
Allow callee to park	Allow called party to park call.
Execute macro	Execute the macro when the called party answers.
Allow caller to hangup	Allow caller to hang up by dialing *.
Allow caller to transfer	Allow caller to transfer the call by hitting the blind xfer keys (features.conf)
Allow caller to record	Allow caller to start recording after pressing *1 or as defined in features.conf.
Allow caller to park	Allow caller to park call.

Property Editor - Audio

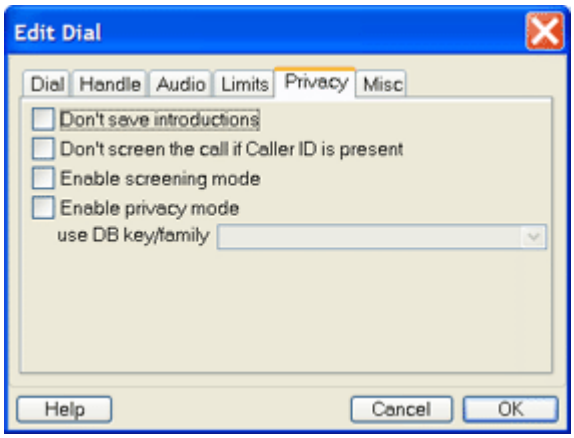


Field	Description
Play announcement to the callee	Play an announcement to the called party.
Send DTMF	After the called party answers, send digits as a DTMF stream, then connect the call to the originating channel. (You can also use 'w' to produce .5 second pauses.)
Play hold music	Provide Music on Hold to the calling party until the called channel answers.
Indicate ringing to the caller	Generate a ringing tone to the calling party. Asterisk will generate a ringing tone automatically if appropriate, even if this option is not selected but with this option selected Asterisk will force ringing tone even if it is not appropriate.

Property Editor - Limits

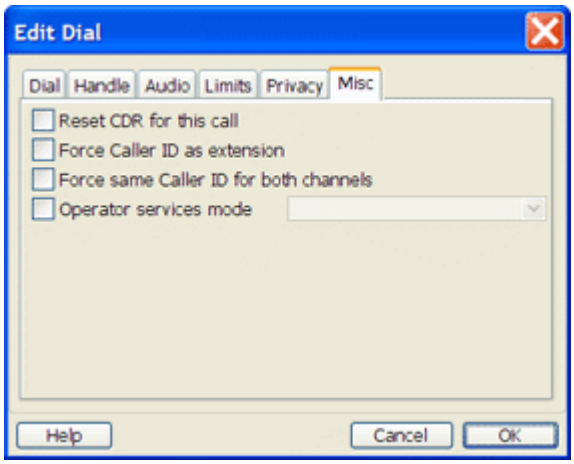
Field	Description
Hangup the call after	Hangup the call after predefined number of seconds, after the called party picks up.
Limit the call to the	Limit the call to predefined number of milliseconds.
play warning when left	Warn when predefined number of milliseconds are left.
repeat warning every	Repeated warning after every predefined number of seconds.
Play sound to the callee	Play sound to the callee.
Play sound to the caller	Play sound to the caller.
connect audio file	Audio file to play when call begins.
warning audio file	Audio file to play as warning. In case the file is not defined, then the default behavior is to announce ("You have [XX minutes] YY seconds").
timeout audio file	Audio file to play when time is up.
Ignore forward requests	Asterisk will ignore any forwarding requests it may receive on this dial attempt.

Property Editor - Privacy



Field	Description
Don't save introductions	Don't save introductions
Don't screen the call if Caller ID is present	This option will disable screening in case the Caller ID is present.
Enable screening mode	This option enables screening mode. This is basic privacy mode. It looks for the file sounds/priv-callerintros/${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:NOCALLERID_${EXTEN}${CUT(CHANNEL,/,1)}=${CUT(CHANNEL,/,2)})}.gsm and if it is not found, prompts the caller to say his name. It then rings the called party and plays sounds/priv-callpending, sounds/priv-callerintros/(see-above), and sounds/screen-callee-options. If the called party enters 1, the call is accepted, 2, the DIAL command exits with ${DIALSTATUS} set to NOANSWER, 3, set to TORTURE and 4, set to DONTCALL. If no valid entry is made, the DIAL command exits with ${DIALSTATUS} set to ANSWER. The check for pre-existence of the name recording may not be what you want. For example, everyone from the same number is not necessarily the same person, especially if the number is OUTOFAREA, but if the file is there, no new name will be recorded. Since the files are never removed, you may wish to remove them with a System(rm /var/lib/asterisk/sounds/priv-callerintros/${IF($["${CALLERID(num)}" != ""]?${CALLERID(num)}:NOCALLERID_${EXTEN}${CUT(CHANNEL,/,1)}=${CUT(CHANNEL,/,2)})}.*) right before the Dial command and clean up old ones with a cron job.
Enable privacy mode	Use the PrivacyManager.

Property Editor - Misc



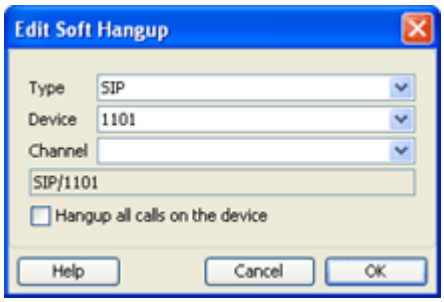
Field	Description
Reset CDR for this call	Reset the CDR (Call Detail Record) for this call.
Force Caller ID as extension	Forces callerid to be set as the extension of the line when making/redirecting the outgoing call. For example, some PSTNs don't allow callerid from other extensions than the ones that are assigned to you.
Force same Caller ID for both channels	This option will force callerid for both, caller and callee.
Operator services mode	This option is valid for Zaptel channel to Zaptel channel only, if specified on non-Zaptel interface, it will be ignored). When the destination answers (presumably an operator services station), the originator no longer has control of their line. They may hang up, but the switch will not release their line until the destination party hangs up (the operator). Specified without an arg, or with 1 as an arg, the originator hanging up will cause the phone to ring back immediately. With a 2 specified, when the "operator" flashes the trunk, it will ring their phone back.

SoftHangup

Request hangup on another channel

This component will hang up the requested channel.

Property Editor



Field	Description
Type	Target device type (SIP, IAX2 etc.)
Device	Target device resource (102, 205 etc.)
Channel	Target channel.
Hangup all calls on the device	Hangup all calls on the selected device instead of resources.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to easier your dialplan development. You can define a connection to Asterisk server in the Preferences dialog.

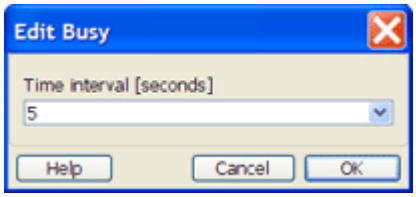
Busy

Indicate busy condition and wait for hangup

This component sends a signal to inform a caller of a channel busy status. The component waits for a caller to hang up and does not continue execution of further components.

Note: This component does not actually play a busy tone to the caller. If you want to play a busy tone, use the Playtones component before the Busy component. Usually the channel or the end device takes care of playing the busy tone to the caller but it is good practice to always use the Playtones component with Busy component.

Property Editor



Field	Description
Time Interval	Hang up the calling channel after the specified number of seconds.

Congestion

Indicate congestion and wait for hang-up

Sends a signal to inform the channel of congestion. This component waits for the caller to hang up and does not continue execution of further components.

Ringing

Indicate ringing tone

This component will indicate the ringing tone to the caller, and will immediately continue the execution of further components.

Example of Usage

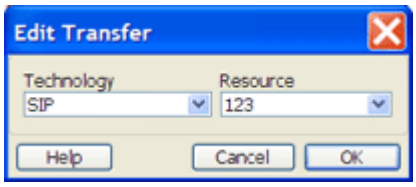
This component is usually used with Wait component to make the caller hear ringing tone for a couple of seconds before he/she is presented with the IVR options.

Transfer

Transfer caller to remote extension

This component will request the remote caller be transferred to a given extension. In case of SIP, IAX2 and LOCAL, only an incoming call with the same channel technology will be transferred.

Property Editor



Field	Description
Technology	Target device technology (SIP, IAX2 etc.)
Resource	Target device resource (102, 205 etc.)

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

Output	Description
Success	Transfer succeeded.
Failure	Transfer failed.
Unsupported	Transfer unsupported by channel driver.

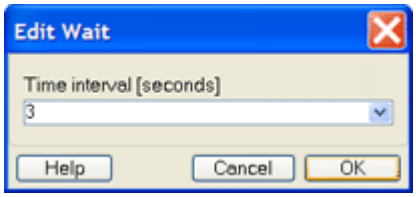
The result of this component will be reported in the TRANSFERSTATUS channel variable.

Wait

Waits for specified time

This component is used to pause a call flow for a specified number of seconds. During the specified time, the sound input received on the channel, including DTMF tones, is silently ignored.

Property Editor



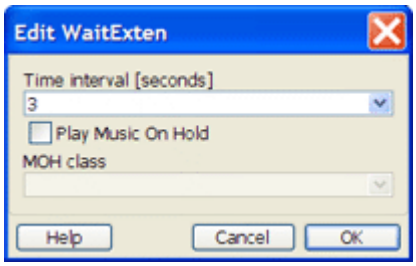
Field	Description
Time interval	A number of seconds to wait.

WaitExten

Wait for the caller to dial new extension

This component will wait for the caller to dial new extension for the specified number of seconds.

Property Editor



Field	Description
Time interval	Number or seconds to wait for the caller to dial new extension. Decimal numbers are accepted (i.e. 1.5 = 1.5 seconds).
Play Music On Hold	Provide music on hold to the caller while waiting for an extension.

Note:

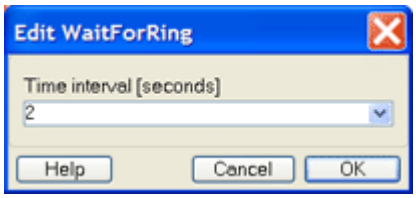
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

WaitForRing

Wait for Ringing component

This component will wait for Ringing component to be executed, for predefined time of seconds, and then will continue with the call flow.

Property Editor



Field	Description
Time interval	Number or seconds to wait. Decimal numbers are accepted (i.e. 1.5 = 1.5 seconds).

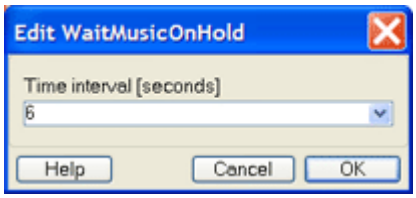
WaitMusicOnHold

Wait and play music on hold

This component will pause a call flow and will play music on hold for specified number of seconds.

Note: In case the music on hold is not available, this component will still pause a call flow for specified number of seconds but without playing any sound.

Property Editor



Field	Description
Time interval	Number or seconds to pause call flow. Decimal numbers are accepted (i.e. 1.5 = 1.5 seconds).

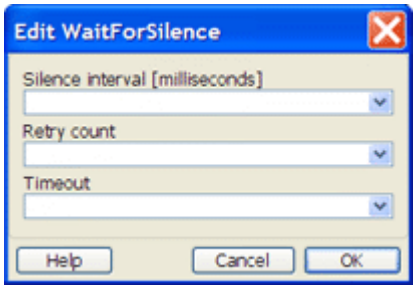
WaitForSilence

Waits for a specified amount of silence

This component will wait for up to specified number of milliseconds of silence (Silence interval), retry specified number of times (Retry count) or once if omitted. An optional timeout specifies the number of seconds to return after, even if we do not receive the specified amount of silence.

Use 'timeout' with caution, as it may change the purpose of this component, which is to wait indefinitely until silence is detected on the line. This is particularly useful for reverse-911-type call broadcast applications where you need to wait for an answering machine. The timeout parameter is specified only to avoid an infinite loop in cases where silence is never achieved. Typically you will want to include two or more calls to WaitForSilence when dealing with an answering machine; first waiting for the spiel to finish, then waiting for the beep, etc.

Property Editor



Field	Description
Silence interval	Silence interval in milliseconds.
Retry count	Number of times.
Timeout	Standard timeout.

ChanIsAvail

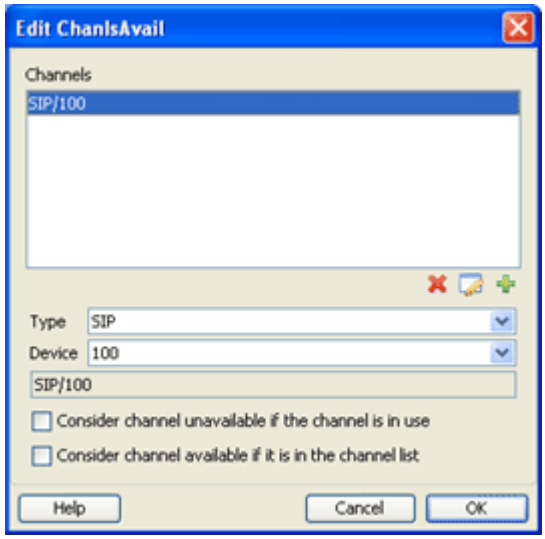
Check if channel is available

Checks if any of the requested channels is available i.e. check if Asterisk can send a call on the channel (but cannot detect if a phone is in use or not).

Note: Currently the ChanIsAvail component works with ZAP, IAX2, mISDN and SIP channels only. MGCP channels are not supported.

Note for SIP, IAX: ChanIsAvail component shouldn't be used to check whether the channel is busy or not, it should be primarily used to check whether it would be possible to send a call on the channel or not. Whether the call would end up being accepted or not is entirely up to the peer that the call is sent to.
The ChanIsAvail component could be also used to determine whether the SIP peer is known and registered, but not for limiting simultaneous calls to the peer.

Property Editor



Field	Description
Channels	A list of channels (given in technology/resource format) to be checked for its availability.
Consider channel unavailable if the channel is in use	Consider the channel unavailable if the channel is in use.
Consider channel unavailable if it is in the channel list	Simply checks if specified channels exist in the channel list.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output options

Output	Description
Available	Some of the requested channels are available.
Unavailable	None of the requested channels is available.

Note: The following variables will be set by this component:
* AVAILCHAN - the name of the available channel, if one exists
* AVAILORIGCHAN - the canonical channel name that was used to create the channel
* AVAILSTATUS - the status code for the available channel (see "devicestate.c")
o 0 AST_DEVICE_UNKNOWN - "Unknown"; channel is valid, but in unknown state.
o 1 AST_DEVICE_NOT_INUSE - "Not in use"
o 2 AST_DEVICE IN USE - "In use"; channel is in use.
o 3 AST_DEVICE_BUSY - "Busy"; channel is busy.
o 4 AST_DEVICE_INVALID - "Invalid"; not known to Asterisk.
o 5 AST_DEVICE_UNAVAILABLE - "Unavailable"; channel is unavailable (not registered)
o 6 AST_DEVICE_RINGING - "Ringing"

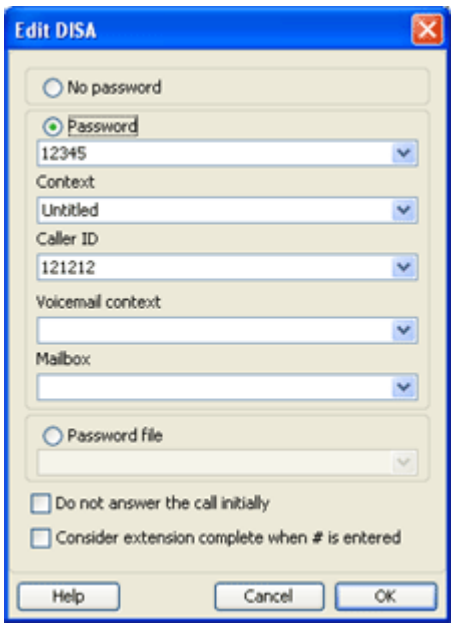
Note that ChanIsAvail component returns not only the name of the channel in AVAILCHAN variable, but also appends the channel's session ID.

DISA

Direct Inward System Access

DISA (Direct Inward System Access) component will provide access to the telephone switch (PBX) from an outside phone line in completely the same way like the call is initiated from an inside extension attached to the switch. Because of the security measures, the component will require the caller to enter a pass code, followed by the pound sign (#). If the pass code is correct, the caller will hear internal switch dial tone and will be available to make calls like he/she is calling from an internal extension. Obviously, this type of access could have serious security implications.

Property Editor



Field	Description
No password	Default option - do not ask for password, just provide access to the switch. Could be serious security issue!
Password	A pass code to be asked before providing access to the switch - single, global pass code for all. It also allows specification of the context on which the caller will be dialing. If no context is specified, the DISA component defaults the context to "disa" presumably that system will have a special context set up for DISA use with some or a lot of restrictions.
Context	Specifies the dialplan context in which the user-entered extension will be matched. If no context is specified, the DISA application defaults the context to "disa". Presumably a normal system will have a

	special context set up for DISA use with some or a lot of restrictions.
Caller ID	Specifies a new (different) callerid to be used for this call.
Voicemail context, Mailbox	This option will cause a stutter-dialtone (indication "dialrecall") to be used, if the specified mailbox contains any new messages.
Do not answer the call initially	The DISA application will not answer the call initially.
Consider extension complete when # is entered	The extension entered will be considered complete when a '#' is entered.
Password file	Individual pass codes contained in a file. The file allows specification of either just a pass code defaulting to the "disa" context, or passcode | context in each line of the file. The file may contain blank lines, or comments starting with "#" or ";". In addition, the above arguments may have | new-callerid-string appended to them, to specify a new (different) callerid to be used for the call, for example: numeric-passcode | context | "My Phone" <(234) 123-4567> - or - full-pathname-of-passcode-file | "My Phone" <(234) 123-4567>

Important notes:

The default options is 'No password' which means that DISA component will provide dial tone without requesting a passcode from the caller. Obviously, this options could be used only if the user's identity is confirmed somewhere in the call flow before DISA component.

Fax

Fax for Asterisk

Fax For Asterisk provides a robust and reliable PSTN and VoIP fax application and modem stack that supports TDM and T.38 endpoints.

Property Editor

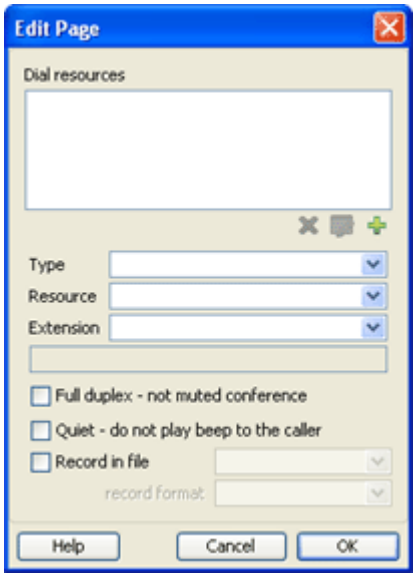
Field	Description
Send or Receive Fax	Select send or receive option
Enable debug reporting	Enable debug reporting
Allow Fallback to audio mode	Allow Fallback to audio mode
Send progress events	Send progress events

Page

Page phones i.e. transmit a message thru multiple phones

This component will initiate outbound call to the given list of phones (in technology/resource format) and will join them into a conference bridge as a muted participants by default. The caller is joined to the conference as a speaker and the conference room is destroyed when the caller leaves the conference. This component requires a working MeetMe installation including an Asterisk timer.

Property Editor



Field	Description
Dial Resources	A list of phones i.e. end points that will be paged in format technology/resource.
Full duplex - not muted conference	Participants will be joined to the conference with a full duplex audio (i.e. not a muted participants).
Quiet - do not play beep to the caller	Do not play beep to the caller.
Record in file	Record the page (conference) to a voice file.

Note:

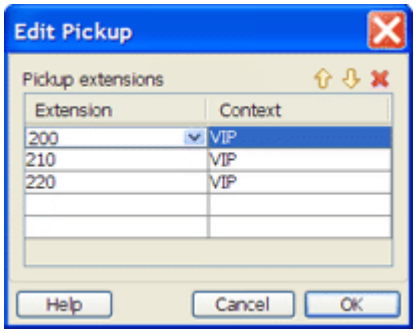
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Pickup

Pickup the ringing channel calling the specified extension

This component will pick up calls ringing on a specific extension. The component also allows partial extensions. For example, you may define a block of extension (200 to 209) and use partial extension number (20 in this case) to pick up most recent call ringing on any of those 10 extensions.
If you use the special string "PICKUPMARK" for the context parameter, for example 10@PICKUPMARK, this component will try to find a channel which has defined a channel variable with the same content as "extension".
Another side effect is that pickup component could be used for outbound calls.

Property Editor



Field	Description
Context	The context to pick up a call from.
Extension	The extension (or partial extension) to pick up a call from.

FollowMe

Find-Me/Follow-Me

This component performs Find-Me/Follow-Me functionality for the caller as defined in the profile matching the <followmeid> parameter in followme.conf. If the specified <followmeid> profile doesn't exist in followme.conf, the call flow will be continued on the next step in the diaplan. Forked i.e. simultaneous dialing of multiple phone numbers in the same step is supported.

Playback category

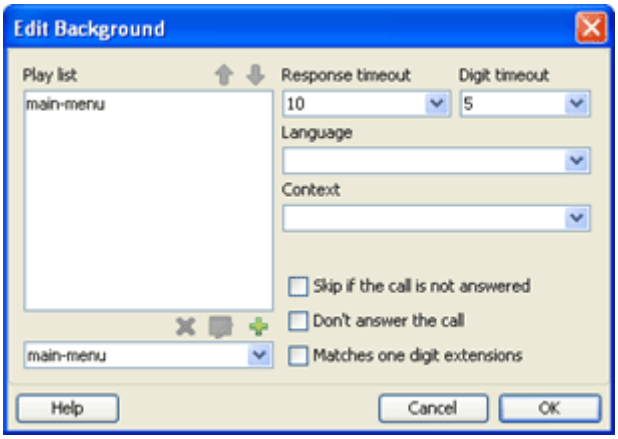
Playback category contains components responsible for playing IVR menus, collecting digits, TTS etc.

Background

Plays a sound file while awaiting caller to enter extension or any other DTMF digits

This component will play the given list of sound files while waiting for extension (or any other digits) to be entered by the calling party. The most common usage of this component is in the IVR menus.
If you want Asterisk to just wait for input without playing a sound file, see the WaitExten component.

Property Editor



Field	Description
Play list	A list of sound files to be played. Expected format of the sound file is 'gsm'.
Response timeout	The maximum amount of time to allow caller to start entering digits (type an extension or similar). If the caller does not start entering digits within this amount of time, dial plan flow will be continued at the 't' extension (if defined) or the call will be terminated in case the 't' extension is not defined.
Digit timeout	Maximum number of seconds between two entered digits i.e. maximum number of seconds to type additional digit before Asterisk considers the input completed. If not specified in the dialplan, the default is 5 seconds. If the caller types a sequence of digits that

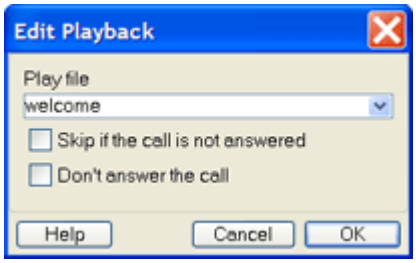
Field	Description
	represents valid extension number or reaches maximum number of expected digits, the input will be interupted immediately, without waiting for the timeout nor '#' sign. Therefore the very fact that a timeout occurs is an indication that the extension number will probably be considered invalid.
Language	Specifies the language to be used for the sound files.
Strategy	Background strategy.
Skip if the call is not answered	In case the channel hasn't been answered the playback will be terminated and the dial plan flow will be continued without paying a sound messages.
Don't answer the call	Don't answer the call before playing the files.
Matches one digit extensions	Continue with the dial plan flow in case the entered digit matches one digit extension in the destination context.
+	move up
-	move down

Playback

Play a sound file

Plays the specified sound file. Sound files are stored in the /var/lib/asterisk/sounds directory by default (the directory path can be changed in asterisk.conf). Comparing to the Background component, which plays a sound file and returns control immediately, Playback will play the whole sound file, and when complete, return the control i.e. continue with the dial plan flow.

Property Editor



Field	Description
Play file	A sound file to play (gsm file is expected). Note that filename extension should be omitted.
Skip if the call is not answered	Play the sound file only if the channel has already been answered. If the channel has not yet been answered, the Playback component will return control immediately without playing the sound file.
Don't answer the call	Play the sound file, but don't answer the channel (if channel hasn't been answered already). Note that some of channels does not support this option.

Output Options

Output	Description
Success	The component played the sound file successfully.
Failed	The component failed to play the sound file.

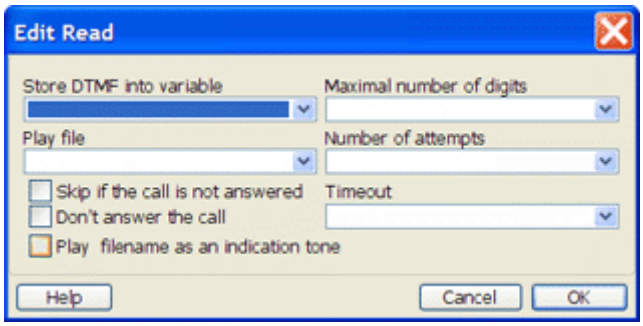
On completion, ${PLAYBACKSTATUS} variable contains either "FAILED" or "SUCCESS".

Read

Collects a digits (DTMF) entered on the channel

This component reads user entered digits terminated by the '#' key and stores the value into a given variable. On error, it sets variable READSTATUS to 'ERROR', which you can catch and handle separately.

Property Editor



Field	Description
Store DTMF into variable	A variable that will store the user entered digits.
Play file	A sound file that will be played before reading a digits.
Maximal number of digits	Maximal acceptable number of entered digits. The component will stop reading digits after this number of digits is reached and will not require user to enter the '#' key. Maximal accepted value is 255. Every value below zero or zero will be treated as a no limit i.e. the component will wait for a user to enter the '#' key.
Number of attempts	The number of attempts to try to read the user entered digits. This value should be greater then 1.
Timeout	This value will override the default timeout if the value is specified and greater than 0.
Skip if the call is not answered	Continue with the dial plan flow and do not read a digits in case the line is not answered.
Don't answer the call	Read a digits even if the call is not answered.
Play filename as an indication tone	Play filename as an indication tone from your indications.conf.

Music On Hold

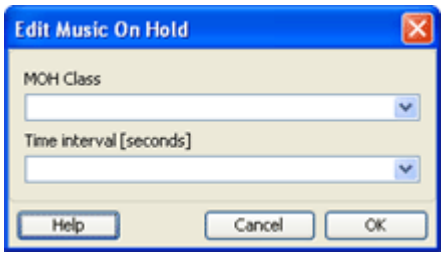
Play Music On Hold

This component plays music on hold specified in musiconhold.conf. If omitted, the default MusicOnHold will be played. The default MusicOnHold can be set using SetMusicOnHold component.

Note:

Remember to answer the line before playing music on hold.

Property Editor



Field	Description
MOH Class	A class defined in musiconhold.conf file.
Time interval	In case the time interval is set, the hold music will be played specified number of seconds. If duration is omitted, music plays indefinitely.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Example of Usage

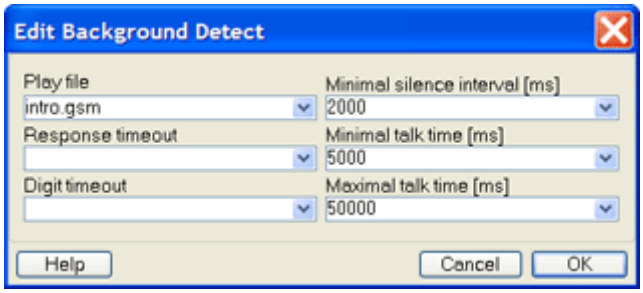
This component could be used to play indefinite message about something, for example about working hours i.e. in case the call arrives between 5PM and 9AM, you could route the call to the MusicOnHold component and notify a caller about working hours.

Background Detect

Plays a background sound and jumps to an extension in case of entered digit or detected caller's voice

BackgroundDetect component is similar to Background component. It plays a given sound file, waiting for a caller to enter digit (entered digit must be first digit of a valid extension, or it will be ignored).
In addition, during the playback of a voice file, the caller's audio input is monitored and in case the period of non-silence is greater than 'Minimal talk time' yet less than 'Maximal talk time' and followed by the silence for at least 'Minimal silance' then the audio playback will be aborted and dial plan flow will be continued at the 'talk' extension (if available).

Property Editor



Field	Description
Play file	A sound file to play. Expected format for the sound file is .gsm.
Response timeout	The maximum amount of time to allow user to start entering digits (type an extension or similar). If the user does not start entering digits within this amount of time, dial plan flow will be continued at the 't' extension (if defined) or the call will be terminated in case the 't' extension is not defined.
Digit timeout	Maximum number of seconds between two entered digits i.e. maximum number of seconds to type additional digit before Asterisk considers this input to be complete. If not specified in the dialplan, the default is 5 seconds. If the user types a sequence of digits that represents valid extension number or reaches maximum number of expected digits, the input will be interpreted immediately, without waiting for the timeout nor '#' sign. Therefore the very fact that a timeout occurs is an indication that the extension number will probably be considered

Field	Description
	invalid when it is interpreted.
Minimal silence interval[ms]	The length of the silance interval after the talking interval (after the caller said something).
Minimal talk time[ms]	Minimal talking time interval that should be detected.
Maximal talk time[ms]	Maximal talking time interval that should be detected.

Example of Usage

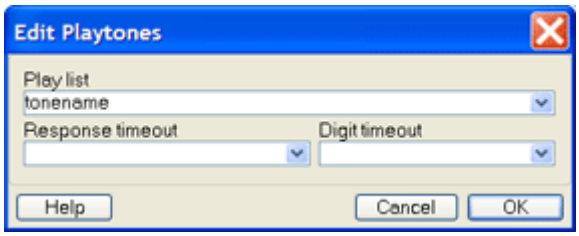
This component could be used for basic answering machine detection.
Basic theory: if there is noise followed by silence within 3 to 5 seconds, assume it's a human ("hello?"), otherwise, wait until the noise stops, and then start leaving a message for a machine. You need to have a couple files called silence/5.gsm and silence/30.gsm in your sounds folder that are just silence.
This may require some tweaking with the timing on BackgroundDetect.

Playtones

Play a tone list

This component plays either the tone name defined in the indications.conf file, or a directly specified tone list of frequencies and durations (see indications.conf for a tone list specification). Dial plan flow will continue with the next step immediately, while the tones continues to play. Use StopPlaytones component to stop the tones playing.

Property Editor



Field	Description
Play list	A tone name defined in the indications.conf file, or a directly specified tone list of frequencies and durations (see indications.conf for a tone list specification).
Response timeout	The maximum amount of time to allow user to start entering digits (type an extension or similar). If the user does not start entering digits within this amount of time, dial plan flow will be continued at the 't' extension (if defined) or the call will be terminated in case the 't' extension is not defined.
Digit timeout	Maximum number of seconds between two entered digits i.e. maximum number of seconds to type additional digit before Asterisk considers this input to be complete. If not specified in the dialplan, the default is 5 seconds. If the user types a sequence of digits that represents valid extension number or reaches maximum number of expected digits, the input will be interpreted immediately, without waiting for the timeout nor '#' sign. Therefore the very fact that a timeout occurs is an indication that the extension number will probably be considered invalid when it is interpreted.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. You can define a connection to the Asterisk server in the Preferences dialog.

Stop Playtones

Stop playing a tone list

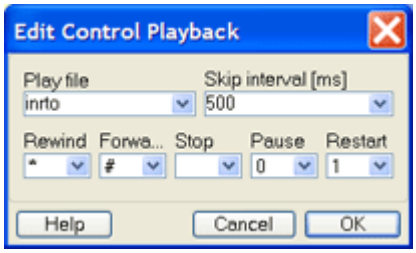
Stops playing a tone name or a tone list that was started by the Playtones component. This component is used in combination with Playtones component.

Control Playback

Plays a sound file and allow fast forward, rewind and stop controls

This component will play a given sound file. The caller may control the playback of the sound by dialing the forward button (default is *) and the rewind button (default is #). Each press of these keys will skip the playback for 'Skip interval' milliseconds. You may also specify stop, restart and pause buttons.

Property Editor



Field	Description
Play file	A sound file to play to the caller. Expected format for the sound file is .gsm (do not include the filename extension).
Skip interval[ms]	Number of milliseconds to skip when rewinding or fast forwarding.
Rewind	Rewind DTMF digit. Rewind when this DTMF digit is received.
Forward	Forward DTMF digit. Forward when this DTMF digit is received.
Stop	Stop DTMF digit. Stop playback when this DTMF digit is received.
Pause	Pause DTMF digit. Pause playback when this DTMF digit is received.
Restart	Restart DTMF digit. Restart playback when this DTMF digit is received.

Output Options

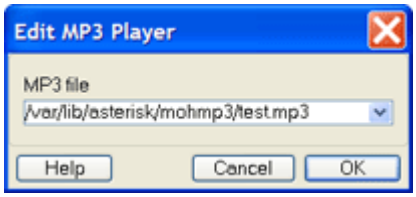
Output	Description
Success	The component executed successfully.
UserStopped	User stopped the playback.
Error	Call processing is terminated (the requested sound files does not exist or channel hangup).

MP3 Player

Plays an MP3 sound file or stream

This component executes the Unix program (mpg123) to play the mp3 file or stream. The caller can exit by typing any digit. MP3 Player component works best with mp3 files without ID3 tags.

Property Editor



Field	Description
MP3 file	The full path to the MP3 file or URL to the MP3 stream.

Example of Usage

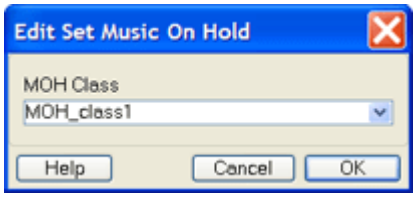
This component is usually used for playing mp3 sound files in the IVR.
Alternatively, it could be used to implement streaming mp3 as standard music-on-hold.

Set Music On Hold

Set default Music On Hold

This component sets the default music on hold for a given channel. When music on hold is activated, this class will be used to select the music to be played.

Property Editor



Field	Description
MOH Class	A class defined in musiconhold.conf file.

Note:

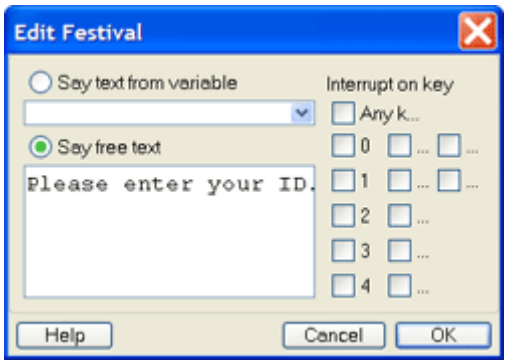
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Festival

Say the text to a caller (text to speech)

This is very powerful component used to generate and play specified text back to the caller. The component requires the Festival open-source speech synthesizer application installed on the system in order to generate the specified text as a sound stream.

Property Editor



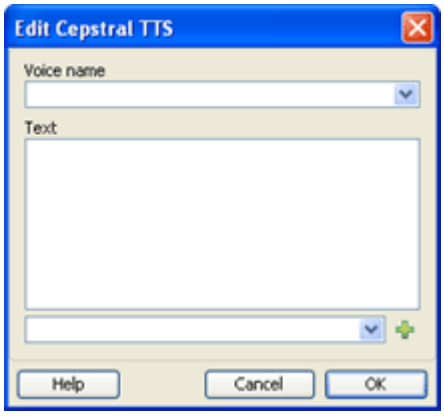
Field	Description
Say text from variable	A variable that contains the text.
Say free text	A text to be played by the component.
Interrupt on key	This filed specifies the buttons that will cause the playing of the sound stream to stop in case it is dialed by the caller. Festival will return the information about the typed button.

Cepstral

Cepstral text to speech conversion and playback (requires Cepstral TTS engine)

This is very powerful component used to generate and play specified text back to the caller. The component requires the Cepstral speech synthesizer application be installed on the system in order to generate the specified text as a sound stream.
The component is located on the Playback sheet.

Property Editor



Field	Description
Voice name	Voice you want to use to speak the text.
Text	A text to be played by the component.

Swift

Speak text through Swift text-to-speech engine

This is very powerful component used to generate and play specified text back to the caller. The component operates in two modes. One is processing text-to-speech while listening for DTMF and the other is just processes the text-to-speech while ignoring DTMF entirely.
The component is located on the Playback sheet.

Property Editor

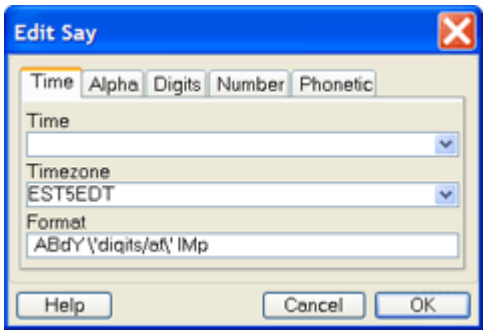
Field	Description
Voice name	Voice you want to use to speak the text.
Listen to DTMF	Listen to DTMF while processing text-to-speech.
Text	A text to be played by the component.

Say

Say time, alpha, digits, number or phonetics.

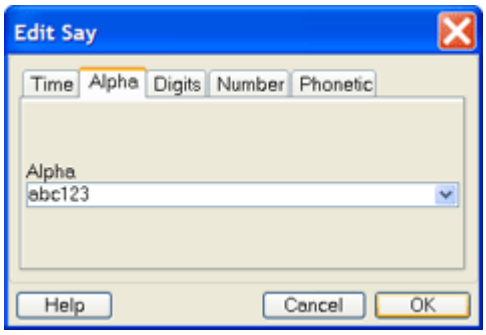
This component says date, time, string of letters, or other characters, digits, number (e.g. "one thousand, two hundred and eighty eight") or phonetics.

Property Editor - Say Time



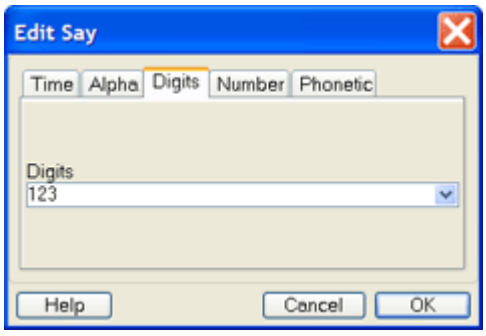
Field	Description
Time	Standard unix timestamp (number of seconds since 1 Jan 1970). If this parameter is omitted, the default value is the current date/time.
Timezone	Specifies the timezone. If this parameter is omitted, the default value will be the time zone of the Asterisk server. The timezone should be specified as a unix timezone (see your /usr/share/zoneinfo directory for a list of timezones known to your computer).
Format	Format is a string that specifies the rule to pronounce the date/time. The default value is: ABdY 'digits/at' IMp This format would result in a phrase: "Monday January twenty first 2003 at seven fifty two p m". Letter — Description A or a — Day of week B or b or h — Month name d or e — A day in the month Y — Year I or l — Hour, 12 hour clock H — Hour, 24 hour clock k — Hour, 24 hour clock M — Minute P or p — AM or PM Q — Date R — 24 hour time, including minute S — seconds

Property Editor - Say Alpha



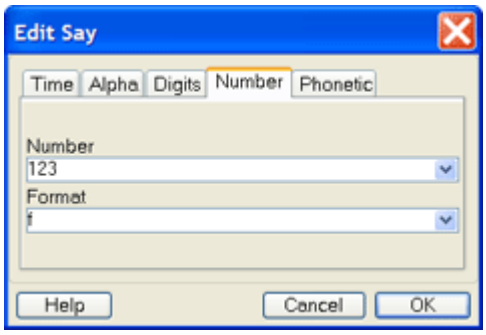
Field	Description
Alpha	A string of letters, numbers, and the following special characters: * ! exclamation point * @ at * # pound * $ dollar * * star * - dash * + plus * = equals * / slash * ' ' (an empty space) space * . dot The component will say each character in the string, one by one, using selected language (default is English).

Property Editor - Say Digits



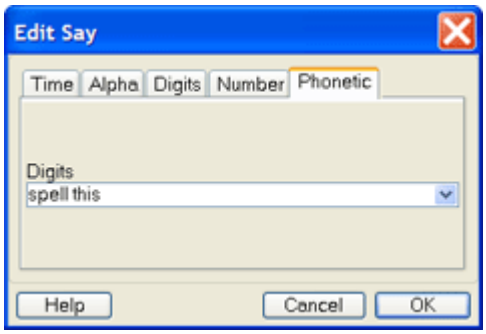
Field	Description
Digits	The digits to be said, one by one, using selected language (default is English).

Property Editor - Say Number



Field	Description
Number	A number to be said, using selected language (default is English). Works only with integer numbers between 0 and 99,999,999.
Format	Gender: "f" female "m" male "c" computer

Property Editor - Say Phonetic



Field	Description
Digits	Characters to pronounce, one by one. This component sequentially plays each file /var/lib/asterisk/sounds/phonetic/character_p.gsm for each character in text. Essentially, this component spells the text using the NATO Phonetic Alphabet.

Progress

Play early audio to the caller before answering the line

This component will request in-band progress information to be provided to the calling channel indicating call progress. This is also known as "early audio" or "early B3".

Example of Usage

This component is typically used in combination with Playback (with 'noanswer' option) to play early audio to the caller (message that will be played before answering the call and starting billing process).

Echo

Echo audio read from the user back to the channel

Echo component will send the outgoing audio stream back to the channel, so the user will hear what he/she said.

Example of Usage

This component is mostly used for test purposes.

Milliwatt

Generate a Constant 1000Hz tone at 0dbm (mu-law)

A "Milliwatt Test Line" is a signal generated from an end office which provides a 1000Hz tone at 0 dBm0 for one-way transmission measurements towards the customer's location.

Example of Usage

This component is usually used for a line testing and measurement purposes.

Integration Server category

This category contains components responsible for communication with Integration Server.

DbQuery

Connects to the database server, executes query, returns result set back and disconnects from the database server

This is extremely powerful component that enables access to the database from the dial plan.
The component requires Integration Server (IS) and can be used to execute SQL statements against any databases that provides JDBC driver. Integration Server out of the box supports the following database servers: MySQL, Microsoft SQL, Sybase, Postgres and JDBC ODBC Bridge.

Here is how it works.

Integration Server (IS) is standalone server application that communicates with Asterisk server through AGI (Asterisk Gateway Interface) and acts as an AGI server that completes AGI requests initiated from the Asterisk dial plan.

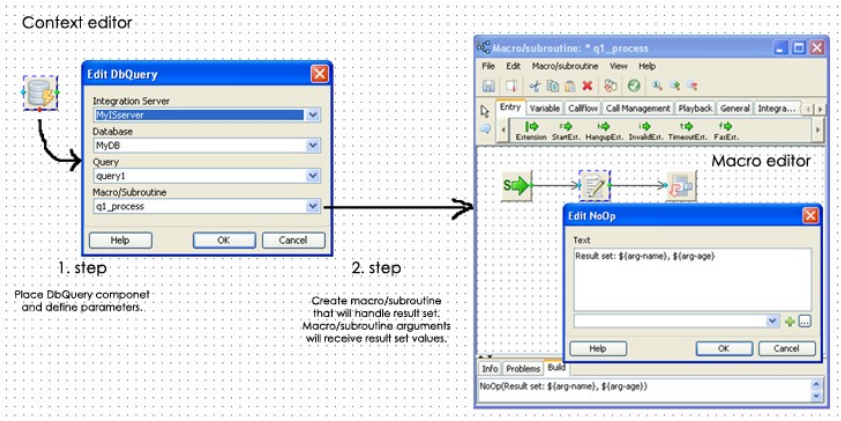
An Asterisk Manager user is required in order to execute AGI calls. Visual Dialplan automatically creates an Asterisk Manager user named is_user with randomly generated password for this purpose. You can later modify both, the Asterisk Manager username and password, if required.

Visual Dialplan deploys traditional extensions.conf code to the Asterisk server. In case the Integration Server functionality is required this code will contain AGI calls to Integration Server. At the same time, Visual Dialplan deploys resources to Integration Server required for those AGI calls.

When new call arrives at the Asterisk server the extensions.conf code (call flow) is executed and AGI request are sent to the IS. IS connects to remote database server, executes SQL queries, returns result set (result of SQL query execution) back to the dial plan and then disconnects from the database server.

More precisely, depending on the selection in the DbQuery component the result set will be stored in the variables (useful in case the result set is one row), processed with the subroutine/macro (in case more than one row is returned from the database) or the result set will not be processed (in case of the update or insert SQL statement).

In case the result set contains several rows the macro/subroutine will be calls several times, as many times as there are returned rows in the result set.



Make sure to define macro/subroutine arguments that will accept SQL query result set values. For example, if you execute the following SQL query:

select name, age from employees

the macro/subroutine should have at least two arguments, one to accept return value for name and the other to accept return value for age. If you define the following two arguments *EmployeeName* and *EmployeeAge* you will have two variables in macro/subroutine named

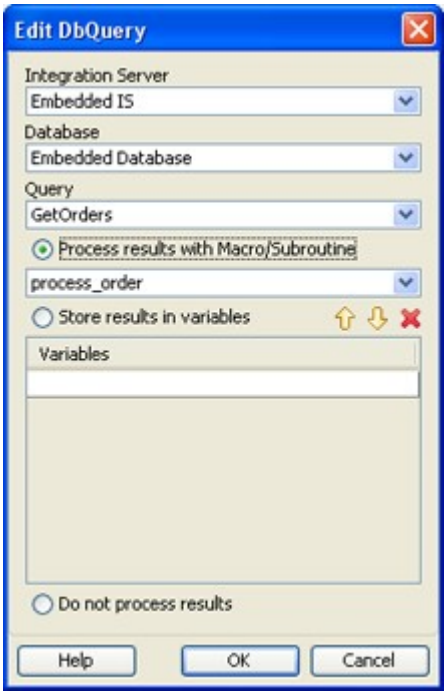
arg-EmployeeName and *arg-EmployeeAge* that will accept return values.

Visual Dialplan automatically creates variables with prefix 'arg-' for entered macro/subroutine parameters.

Note:

- Integration Server and the connection to database server should be configured properly in order to use DbQuery component
- Make sure to set the database remote access privileges properly in order to access the database server from Integration server
- Make sure to define macro/subroutine arguments that will accept SQL query result set values

Property Editor



Field	Description
Integration Server	Select one of defined Integration Servers at the Integration Server view in Visual Dialplan.
Database	Select one of database server connections (under Database Resources) defied for selected Integration Server.
Query	Select one of defined SQL queries (under Queries) defined for selected database server connection.
Process results with Macro/Subroutine	Select Macro/Subroutine that is responsible for handling SQL query results. Make sure to define macro/subroutine arguments that will accept SQL query result set values. For example, if you execute the following SQL query: select name, age from employees the macro/subroutine should have at least two arguments, one to accept return value for name and the other to accept return value for age. If you define the following two arguments EmployeeName and EmployeeAge you will have two variables in macro/subroutine named arg-EmployeeName and arg-EmployeeAge that will accept return values. In case the SQL query returns one row the macro/subroutine will be called one time, but if SQL query

	returns 2 or more rows the macro/subroutine will be called as many times as the number of return rows is.
Store results in variables	Select variables that will store the SQL query result values. For example, if you execute the following SQL query: select name, age from employees where id = 1 the variable table should have two variables, one to accept return value for name and the other to accept return value for age. Option "Store results in variables" should be used only when SQL query returns only one row of data. If SQL query returns 2 or more rows the macro/subroutine will be called only for the first row of data.
Do not process results	Select this option if results should not be processed (e.g. insert, update or delete SQL statements).

Note:

You must create macro/subroutine manually, as well as the macro/subroutine arguments.

Note:

Integration Server creates IS_DB_RESULT_INDEX variable and sets its velue to the current number of the row in the result set. For example, if the result set contains 3 rows, the IS_DB_RESULT_INDEX variable will be one when subroutine/macro works with the first row of the result set, then value two when subroutine/macro works with the second row and lastly value three when subroutine/macro works with the third row of the result set.

Note:

When working with update statement, Integration Server creates IS_DB_UPDATE_COUNT variable and sets its value to the current number of updated rows.

SendEmail

Connects to SMTP server and sends email

This is powerful component that work with emails from the dial plan.
The component requires Integration Server (IS).

Here is how it works.
Integration Server is standalone server side application that communicates with Asterisk server through AGI (Asterisk Gateway Interface) and acts as an AGI server that completes AGI requests initiated from an Asterisk dial plan. Visual Dialplan deploys traditional extensions.conf code at Asterisk server and email resources at Integration Server (email template and similar).

When a new call arrives at the Asterisk server the extensions.conf code (call flow) is executed and AGI request is sent to IS. IS connects to remote SMTP (email) server, authenticates and sends email.

Note:

Integration Server and connection to SMTP (email) server should be configured properly in order to use SendMail component.

All variables (e.g. ${variable}) will be replaced with its values.

For example, HTML email template, with *name* variable, and predefined value to *Michael* , for the email preview/test purpose only (in run time variable *name* will be replaced with the value of *name* dial plan channel variable), may look like this:

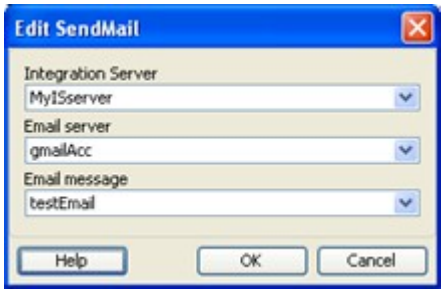
<!@@ ${name}=Michael>
<h1>Test Mail</h1>

Hello ${name},

This is test email.

Thank you

Property Editor



Field	Description
Integration Server	Select one of defined Integration Servers at the Integration Server view in Visual Dialplan.
Email Server	Select one of SMTP (email) server connections (under Email Resources) defied for selected Integration Server.
Email message	Select one of defined email templates (under Templates) defined for selected email server connection.

ProcessPayment

Connects to payment server and process payments.

This is powerful component that process payments from the dial plan. The component requires Integration Server (IS).

Here is how it works.

Integration Server is standalone server application that communicates with Asterisk server through AGI (Asterisk Gateway Interface) and acts as an AGI server that completes AGI requests initiated from an Asterisk dial plan.

Visual Dialplan deploys traditional extensions.conf code at Asterisk server and payment resources at Integration Server.

When a new call arrives at the Asterisk server the extensions.conf code (call flow) is executed and AGI request is sent to IS. IS connects to remote payment server, authenticates and process payment.

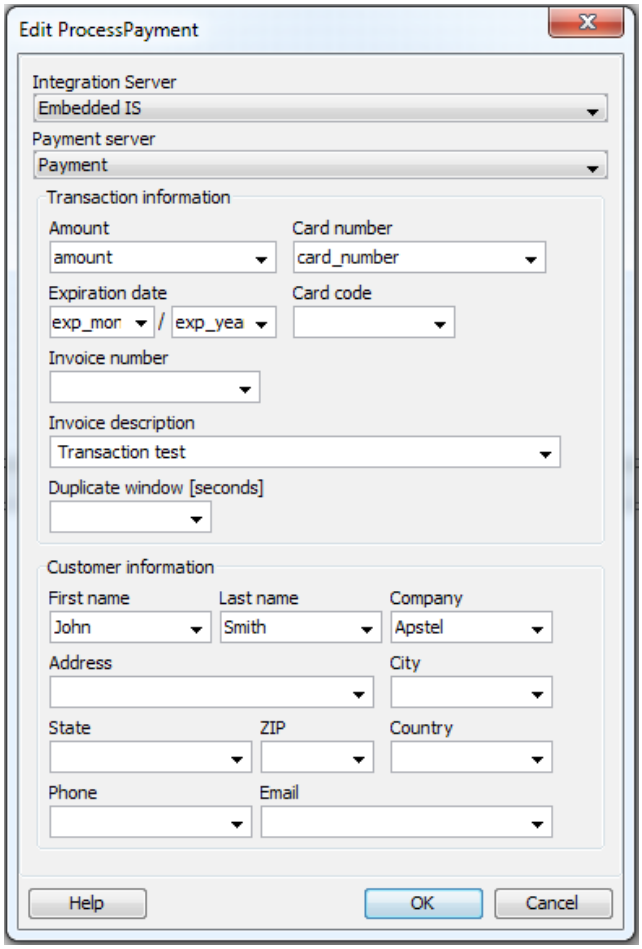
The component is located on the Integration Server sheet.

Note:

Integration Server and connection to processor server should be configured properly in order to use ProcessPayment component.

All variables (e.g. ${variable}) will be replaced with its values.

Property Editor



Field	Description
Integration Server	Select one of defined Integration Servers at the Integration Server view in Visual Dialplan.
Payment Server	Select one of Payment server connections (under Payment Resources) defied for selected Integration Server.
Transaction information	This section takes transaction informations such as: Amount, Card number, Expiration date, Card code, Invoice number, Invoice description, Duplicate window
Customer information	This section takes customer informations such as: First name, Last name, Company, Address, City, State, ZIP, Country, Phone, Email.

General category

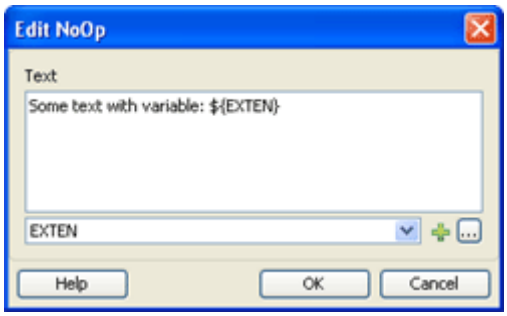
The General category contains components responsible for some commonly used tasks like logging, setting verbose level, sending DTMF etc.

NoOp

No operation

This component will write a text to the command line interface (CLI).

Property Editor



Field	Description
Text	A text to write to the CLI.

Example of Usage

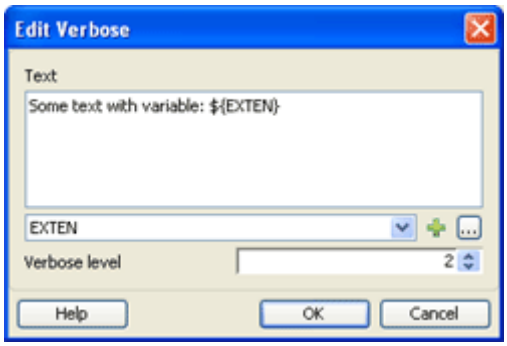
NoOp component could be used for dial plan debugging purposes. For example, Zap channels do not print the callerid information on incoming calls and this component could be used to print it. Please note that you should specify verbose level to 3 or higher in order to get some useful results from this component.

Verbose

Writes a text to the command line interface (CLI) with specified verbose level

This component will write a text to the command line interface (CLI) using verbose message system. This is similar in effect to the NoOp component, but NoOp only output when the verbosity level is 3 or more.

Property Editor



Field	Description
Text	A text to write to the CLI.
Verbose level	Define the verbose level (1-4). In case the verbose level is less than 1 or greater than 4, the verbosity will be 1. If not specified, the verbose level will be 0.

Example of Usage

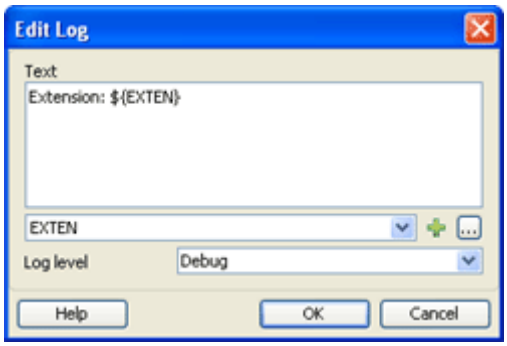
The Verbose component could be used for dial plan debugging purposes. For example, Zap channels do not print the callerid information on incoming calls and this component could be used to print it.

Log

Send arbitrary text to a selected log level

This component may be used to send arbitrary text to a selected log level.

Property Editor



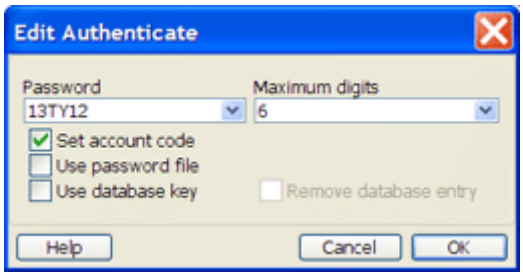
Field	Description
Text	Arbitrary text.
Log level	The level may be one of the following: ERROR, WARNING, NOTICE, DEBUG, VERBOSE, DTMF.

Authenticate

Authenticate a user with a dialed pass code

This component requires a caller to enter a password in order to continue with the dial plan flow; otherwise the flow will be terminated. The component is usually used to restrict the access to the telephony system.

Property Editor



Field	Description
Password	Based on selected options this filed may contain one of the following values: authentication code digits, full path to the password file or database key.
Set account code	If this option is checked, the password would be stored in the CDR field "accountcode" and the channel variable ${ACCOUNTCODE} would be set.
Use password file	Use this option if you provided a full path to the password file in the 'Password' box. Password file should contain one password per line. Usernames or channels cannot be specified in the password file.
Use database key	Use this option if you provided a database key in the 'Password' drop-down box.
Remove database entry	Remove a database key after the successful entry (only applicable together with 'Use database key' option).
Maximum digits	Maximum acceptable number of digits. Stops reading after this number of digits have been entered. Defaults to 0 i.e. no limit (wait for the user to press the '#' key).

Example of Usage

Authenticate component could be used to create private sections in the IVR menu. You may decide to have public part of the IVR, available to all callers, and a private part that will require authentication code (for example the sections where caller may check its bills or inquire invoices).

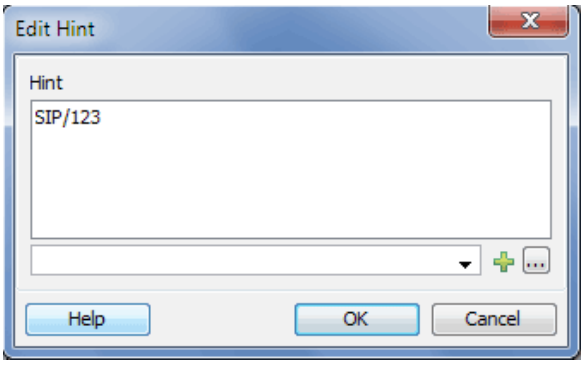
Hint

Add dialplan hints

This component will add dial plan hints. Hints are used to associate an extension with Asterisk channel for the purpose of mapping a state of the channel to a state of the extension.

The component is located on the General sheet.

Property Editor



Field	Description
Hibt	A hint to add to the dialplan.

Example of Usage

If you want to monitor a state of multiple phones using one speed dial, you can place Hint component with hint:

SIP/201&SIP/202&SIP/203

In Asterisk 1.6 and higher, it is possible to use dynamic hints:

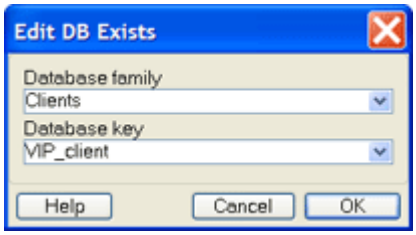
SIP/${EXTEN}

Ast DB Exists

Branches based on database key existence

This component checks if a key exists in the Asterisk database and continues dial plan flow accordingly.

Property Editor



Field	Description
Database family	Asterisk database family.
Database key	Asterisk database key.

Output Options

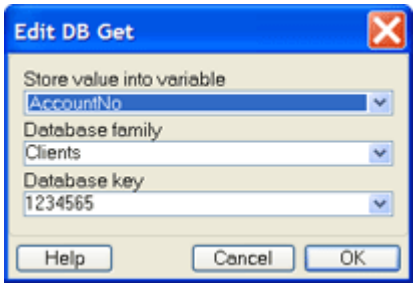
Output	Description
Success	Database key exists.
Error	Database key does not exist.

Ast DB Get

Read a value from the database

Retrieves a value from the Asterisk database and stores it in the given variable.

Property Editor



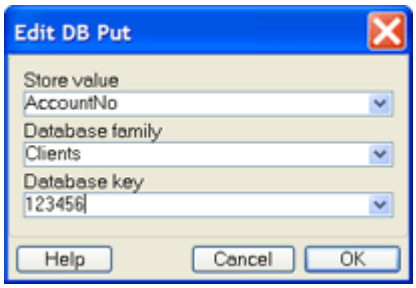
Field	Description
Store value into variable	The variable to store the value to.
Database family	Asterisk database family.
Database key	Asterisk database key.

Ast DB Put

Write a value to the database

Stores the value in the Asterisk database.

Property Editor



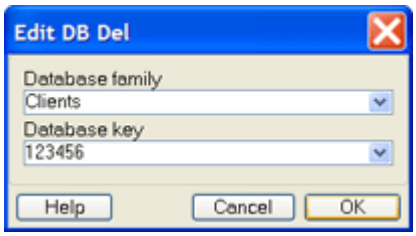
Field	Description
Store value	The value to store to database.
Database family	Asterisk database family.
Database key	Asterisk database key.

Ast DB Del

Delete a key from the database

Removes a key from the Asterisk database.

Property Editor



Field	Description
Database family	Asterisk database family.
Database key	Asterisk database key.

Ast DB Del Tree

Delete a family or key tree from the database

Removes a family or key tree from the Asterisk database.

Property Editor

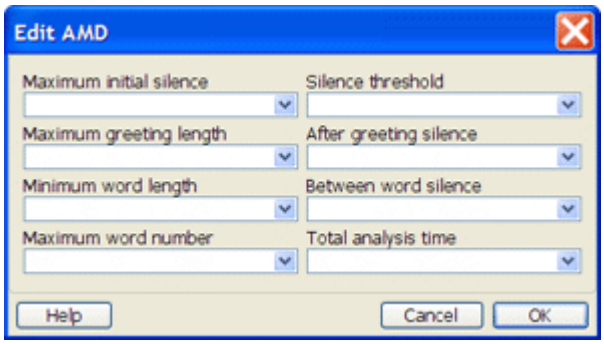
Field	Description
Database family	Asterisk database family.
Database key	Asterisk database key.

Amd

Attempts to detect answering machines

This component attempts to detect answering machines at the beginning of an outbound calls. Simply call this component after the call has been answered (outbound calls only). When loaded, AMD reads amd.conf and uses the parameters specified as default values. Those default values are overwritten with provided parameters.

Property Editor



Field	Description
Maximum initial silence	Maximum initial silence duration before the greeting.
Maximum greeting length	Maximum length of a greeting.
Minimum word length	The minimum duration of Voice to be considered as a word.
Maximum words number	The maximum number of words in the greeting.
Silence threshold	This is a silence threshold.
After greeting silence	The silence after detecting a greeting.
Between word silence	The minimum duration of silence after a word to consider the audio that follows as a new word.
Total analysis time	The maximum time allowed for the algorithm to decide between the HUMAN and MACHINE.

Custom Code

This component allows user to write traditional Asterisk dialplan code and to keep that code inside the Visual Dialplan. That way user can develop call flow in Visual Dialplan and still keep part of the traditional dial plan code and continue to maintain that code inside Visual Dialplan.

This component can be also used to execute some newly added functionality in Asterisk that is still not fully implemented in Visual Dialplan.

Example of Usage

This component can be very powerful when moving your existing Asterisk dialplan, developed in traditional way, to Visual Dialplan.

Exe

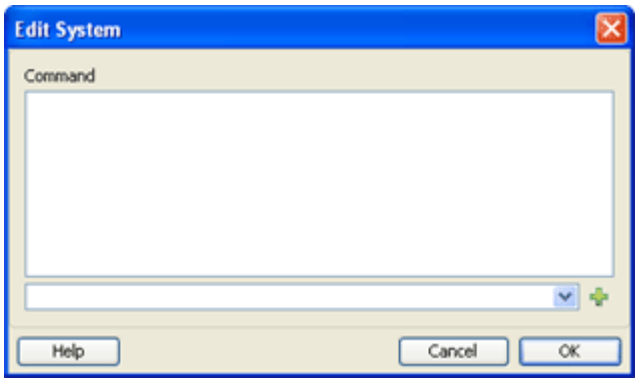
Exe category contains components responsible for integration with other applications (AGI, external IVR etc.).

System

Execute a system (Linux shell) command

This component will execute the Linux shell command from within the dial plan flow.

Property Editor



Field	Description
Command	A Linux shell command with arguments.

Output Options

Output	Description
Success	The Linux shell command executed successfully.
Failure	The Linux shell could not execute the specified command.
Application error	The Linux shell command returned the application error.

Example of Usage

This component could be used to restart the Asterisk PBX by a simple phone call.

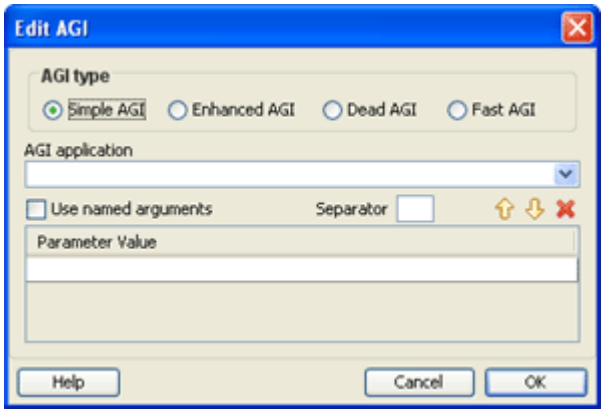
Agi

Executes an AGI compliant application

This component will execute an AGI (Asterisk Gateway Interface) compliant program on a channel. AGI allows Asterisk to launch external programs to control a telephony channel, play audio, read DTMF digits, etc. by communicating with the AGI protocol on stdin and stdout. Requirements for applications

- must be executable
- must be located in /var/lib/asterisk/agi-bin

Property Editor



Field	Description
Simple	Choose this for standard AGI application.
Enhanced	Enhanced AGI provides audio available out of band on file descriptor 3.
Dead	Use this option to execute an AGI script in the 'h' hangup extension.
Fast	Fast AGI allows you to run AGI remotely over TCP socket (agi://xxx).
Use named arguments	Enables to defines arguments in for key=value for Simple, Enhanced and Dead AGI types.
Separator	Define separator for arguments. The default value for Simple, Enhanced and Dead AGI types is space, and for Fast AGI type the default value is &.
AGI application	The AGI application.
Parameter value	The AGI application expected parameters.

ExternalIvr

Executes an external IVR generator

This component implements a simple protocol for bidirectional communication with an external process, while simultaneous playing audio files to the connected channel (without interruption or blocking). The arguments to ExternalIVR component consist of the command to execute and any arguments to pass to it, the same as the Asterisk cmd System application accepts. The external command will be executed in a child process, with its standard file handles connected to the Asterisk process as follows:
stdin (0), DTMF and hangup events will be received on this handle
stdout (1), Playback and hangup commands can be sent on this handle
stderr (2), Error messages can be sent on this handle

The application will also create an audio generator to play audio to the channel, and will start playing silence. When your application wants to send audio to the channel, it can send a command (see below) to add file(s) to the generator's playlist. The generator will then work its way through the list, playing each file in turn until it either runs out of files to play, the channel is hung up, or a command is received to clear the list and start with a new file. At any time, more files can be added to the list and the generator will play them in sequence.

While the generator is playing audio (or silence), any DTMF events received on the channel will be sent to the child process (see below). Note that this can happen at any time, since the generator, the child process and the channel thread are all executing independently. It is very important that your external application is available to receive events from Asterisk at all times (without blocking), or the application could cause the channel to become non-responsive. If the child process dies, ExternalIVR component will notice this and hang up the channel immediately (and also log a corresponding message).

The component is located on the Integration sheet.

DTMF (and other) events

All events will be newline-terminated strings. Events send to the child's stdin will be in the following format:
tag,timestamp[,data]

The tag can be one of the following characters:
0-9, DTMF event for keys 0 through 9
A-D, DTMF event for keys A through D
*, DTMF event for key *
#, DTMF event for key #
H, the channel was hung up by the connected party
Z, the previous command was unable to be executed (file does not exist, etc.)
T, the play list was interrupted (see below)
D, a file was dropped from the play list due to interruption (the data element will be the dropped file name)
F, a file has finished playing (the data element will be the file name)

The timestamp will be 10 digits long, and will be a decimal representation of a standard Unix epoch-based timestamp.

Commands

All commands must be newline-terminated strings. The child process can send commands on stdout in the following formats:
S,filename
A,filename
H,message
O,option

The 'S' command checks to see if there is a playable audio file with the specified name, and if so, clears the generator's playlist and places the file onto the list. Note that the playability check does not take into account transcoding requirements, so it is possible for the file to not be played even though it was found. If the file cannot be found, a 'Z' event (see above) will be sent to the child. If the generator is not currently playing silence, then T and D events will be sent to the child to signal the playlist interruption and notify it of the files that will not be played.

The 'A' command checks to see if there is a playable audio file with the specified name, and if so, adds it to the generator's playlist. The same playability and exception rules apply as for the 'S' command.

The 'H' command stops the generator and hangs up the channel, and logs the supplied message to the Asterisk log.

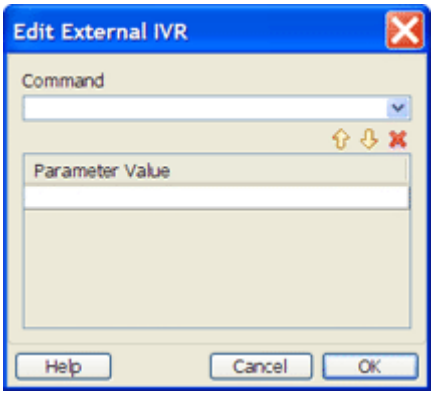
The 'O' command allows the child to set/clear options in the ExternalIVR() application. The supported options are: autoclear, Automatically interrupt and clear the playlist upon reception of DTMF input.

noautoclear, Do not automatically interrupt and clear the playlist upon reception of DTMF input.

Errors

Any newline-terminated output generated by the child process on its stderr handle will be copied into the Asterisk log.

Property Editor



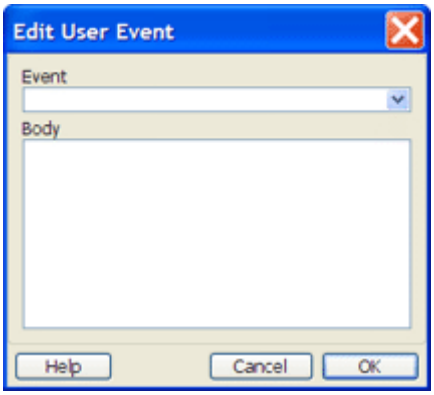
Field	Description
Command	External program or a Linux shell command.
Parameter Value	A list of arguments/parameters.

UserEvent

Send an arbitrary event to the manager interface

This component will send an arbitrary event to the manager interface, with an optional body representing additional arguments. The format of the event will be:
Event: UserEvent < specified event name >
Channel: < channel name >
Uniqueid: < call uniqueid >
[body]

Property Editor



Field	Description
Event	This field specifies the user event name.
Body	This is optional field. If the body is not specified, only Event, Channel, and Uniqueid fields will be present.

Ast Addons

This category contains components responsible for communication with internal Asterisk database.

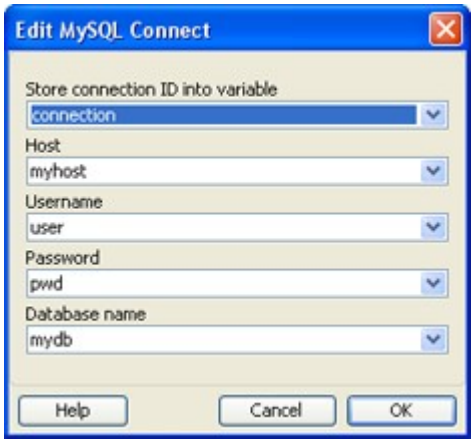
MySql Connect

Connect to MySql database

Connects to a MySql database. Arguments contain standard MySql parameters passed to a function mysql_real_connect.

Note: The component is part of the Asterisk add-on package and will work only if Asterisk add-on package is installed on Asterisk box.

Property Editor



Field	Description
Store connection ID into variable	Variable used to store connection ID obtained from the database.
Host	Database host/server to connect to.
Username	Username used to connect to the database.
Password	Password used to connect to the database.
Database name	Name of the database to connect to.

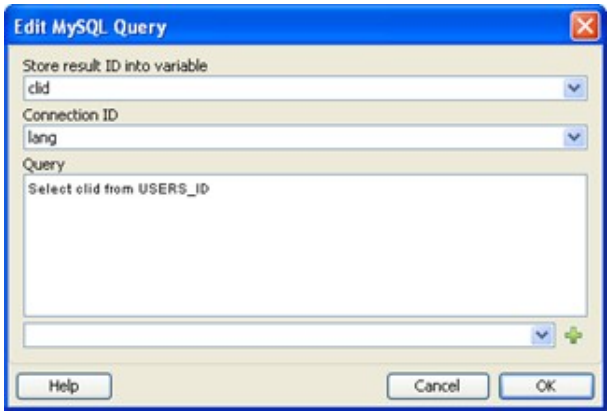
MySql Query

Executes standard MySql query

Executes standard MySql query contained in query string, using established connection obtained by the call to the MySql Connect component.

Note: The component is part of the Asterisk add-on package and will work only if Asterisk add-on package is installed on Asterisk box.

Property Editor



Field	Description
Store result ID into variable	Specify variable to store the result ID, so that it can be used later by MySql Fetch component.
Connection ID	Connection ID obtained from the database when a connection was established.
Query	Standard MySql query that should be executed. Asterisk variables can also be added into the query. Note that variable values will be replaced into the query string during the runtime.

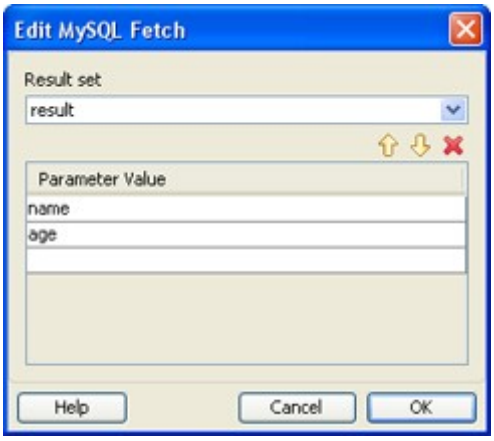
MySql Fetch

Fetch a single row from a query result set

Fetches a single row from a previously executed query and stores the data in the specified variables.

Note: The component is part of the Asterisk add-on package and will work only if Asterisk add-on package is installed on Asterisk box.

Property Editor



Field	Description
Result set	Result set ID obtained by the call to the MySql Query component.
Variables	List of variables used to store fetched data.

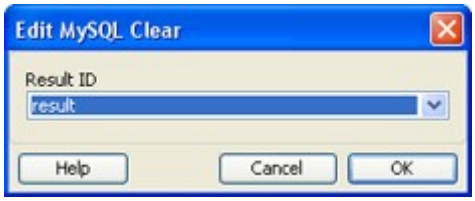
MySql Clear

Free result set memory

Frees memory and data structures associated with result set.

Note: The component is part of the Asterisk add-on package and will work only if Asterisk add-on package is installed on Asterisk box.

Property Editor



Field	Description
Result set	Result set ID obtained by the call to the MySql Query component.

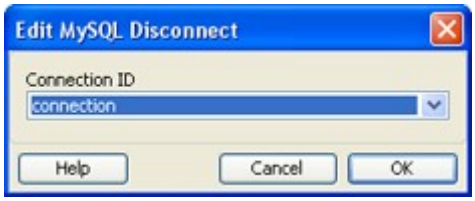
MySql Disconnect

Disconnect from MySql database

Disconnects from the MySql database to which the connection was obtained by the call to the MySql Connect component.

Note: The component is part of the Asterisk add-on package and will work only if Asterisk add-on package is installed on Asterisk box.

Property Editor



Field	Description
Connection ID	Connection ID obtained from the database when a connection was established.

VM & Conf category

The Voicemail and Conferencing category contains components responsible for voicemail and voice conferencing management.

Directory

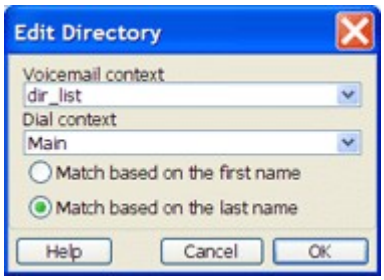
Provides directory of voicemail extensions

This component will present caller with a directory of extensions from which the caller may select by name. This feature is known as 'Dial by name' on other PBX systems. The list of names and extensions is discovered from the voicemail.conf file.

The dialog flow:

- Plays directory introduction file (dir-intro) and waits up to 5 seconds to read caller input (3 digits)
- Intro file says "Please enter the first three letters of the persons last name..."
- Name is the last word found in the 'name' field in the voice mailbox entry in voicemail.conf file
- Plays directory instructions file (dir-instr) for instructions on how to connect to that extension.
- Also plays the "name" as recorded by the voice mailbox owner to identify the extension. If this recording does not exist, it will speak the letters of the name (bee-oh-bee-space-ess-em-aye-tee-aich)
- If more than one matching last name is found, it will allow the caller to cycle through all the matches found
- If no matches, it repeats the introduction
- Pressing "*" will exit
- Pressing "1" will exit setting up the channel to enter the extension selected

Property Editor



Field	Description
Voicemail context	The context in which to interpret the extensions.
Dial context	The context to use for dialing the users. Default is vm-context.
Match based on the first name	Match caller entered digits (3 digits) to the first name (as defined in the voicemail.conf file)
Match based on the last name	Match caller entered digits (3 digits) to the last name (as defined in the voicemail.conf file). This is the default option.

Note:

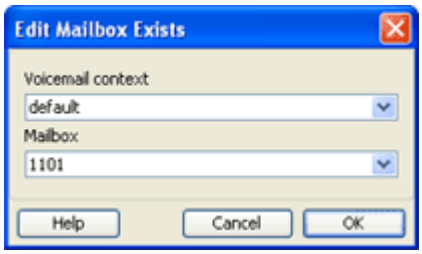
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

MailboxExists

Checks if voice mailbox exists

This component will branch call flow based on the voice mailbox existence.

Property Editor



Field	Description
Voicemail context	The context in voicemail.conf file in which the voice mailbox is created. If no context is specified, the 'default' context will be used.
Mailbox	The voice mailbox to check against its existence.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

Output	Description
Success	The requested voice mailbox exists.
Failed	The requested voice mailbox does not exist.

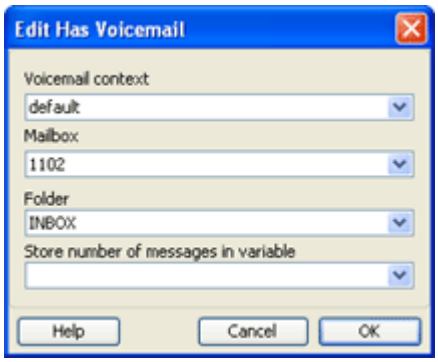
This component will set the following channel variable upon completion: VMBOXEXISTSSTATUS
Possible values are SUCCESS or FAILED.

HasVoicemail

Branches if there are new voicemail messages in the folder

This component is used to branch the call flow based on the status of new voicemail messages in the specified voicemail box and folder.

Property Editor



Field	Description
Voicemail context	The context in voicemail.conf in which the voice mailbox is created. This is optional argument.
Mailbox	The mailbox to check against new voicemails.
Folder	The particular folder in the maibox above that you want to check against new voicemails. The default is INBOX.
Store number of messages in variable	The variable that will store the number of new voicemail messages.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

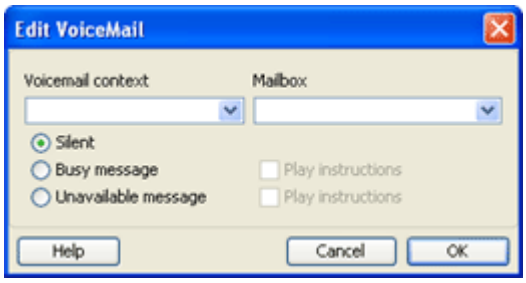
Output	Description
Success	There are new voicemails in the specified voicemail box and folder.
No voicemail messages	There are no new voicemails in the specified voicemail box and folder.

VoiceMail

Leave a voicemail message

This component will record a voicemail message (record a channel) and will save it as an audio file in a given voice mailbox (that must be configured in the voicemail.conf file).

Property Editor



Field	Description
Voicemail context	The context in voicemail.conf in which the voice mailbox is created.
Mailbox	The mailbox to leave the voicemail message to.
Silent	If selected the instructions will be skipped ("Please leave your message after the tone. When done, hang up, or press the pound key.").
Busy message	Play busy message. By default, the message says, "The person at extension < extension number > is busy". If 'Play instruction' option is checked, the instruction will be played too, otherwise only the busy message would be played.
Unavailable message	Play unavailable message. By default, the message says, "The person at extension < extension number > is unavailable", but the mailbox owner may record his/her own unavailable message with the VoicemailMain component. If 'Play instruction' option is checked, the instruction will be played too, otherwise only the unavailable message would be played.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

Output	Description
Success	Voicemail message was left successfully.
User exit	User pressed '0' (zero) during the announcement and continued with the call flow (the option 'operator=yes' should be set in voicemail.conf file.
Failed	Voicemail message was not left successfully.

This component will set the following channel variable upon completion: VMSTATUS

VoicemailMain

Enter voicemail system

This is very powerful component, it will allow the complete voice mailbox management through IVR system. The mailbox can be passed as an argument to this component and in that case the system will not prompt for the mailbox.

The IVR menu:

1 Read voicemail messages

3 Advanced options

1 Reply

2 Call back(1)

3 Envelope

4 Outgoing call(1)

4 Play previous message

5 Repeat current message

6 Play next message

7 Delete current message

8 Forward message to another mailbox

9 Save message in a folder

* Help; during msg playback: Rewind

Exit; during msg playback: Skip forward

2 Change folders

0 Switch to new Messages

1 Switch to old Messages

0 Mailbox options

1 Record your unavailable message

2 Record your busy message

3 Record your name

4 Record your temporary

5 Change your password

* Return to the main menu

* Help

Exit

After recording a message (incoming message, busy/unavailable greeting, or name)

1 - Accept

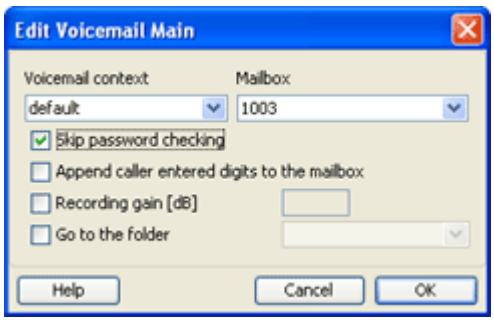
2 - Review

3 - Re-record

0 - Reach operator(1) (not available when recording greetings/name)

(1) Prompts for these are only played if these options are enabled in voicemail.conf

Property Editor



Field	Description
Voicemail context	The context in voicemail.conf in which the voice mailbox is created. This is optional argument.
Mailbox	The voice mailbox to manage.
Skip password checking	Skip checking the passcode for the mailbox.
Append caller entered digits to the mailbox	Consider the mailbox parameter as a prefix to the mailbox that is entered by the caller.
Recording gain [dB]	Use the specified amount of gain when recording a voicemail message. The units are whole-number decibels (dB).
Go to the folder	Skip folder prompt and go directly to the specified folder. Default is Inbox folder.

Note:

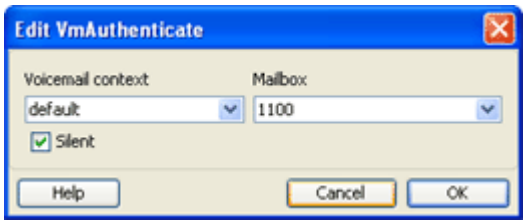
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

VmAuthenticate

Authenticate a user based on voicemail.conf

This component behaves identically to the Authenticate component, with the exception that the passwords are taken from the voicemail.conf file.

Property Editor



Field	Description
Voicemail context	The context in voicemail.conf in which the voice mailbox is created.
Mailbox	If the mailbox is specified, only that mailbox's password will be considered valid. If the mailbox is not specified, the channel variable AUTH_MAILBOX will be set with the authenticated mailbox.
Silent	Initial prompts will not be played if this field is checked.

Note:

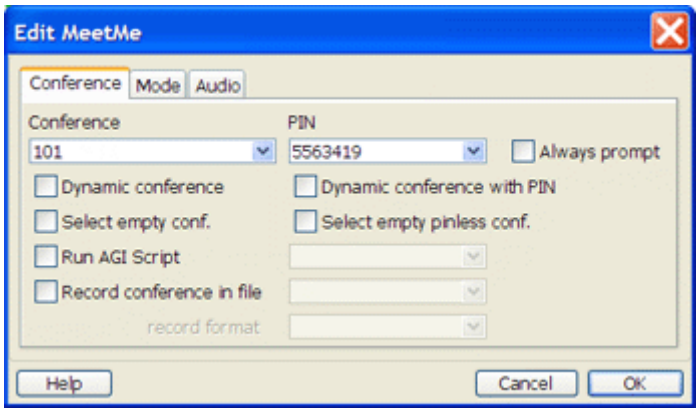
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

MeetMe

Simple MeetMe conference bridge

This component will enter the caller into a specified MeetMe conference room.

Property Editor - Conference



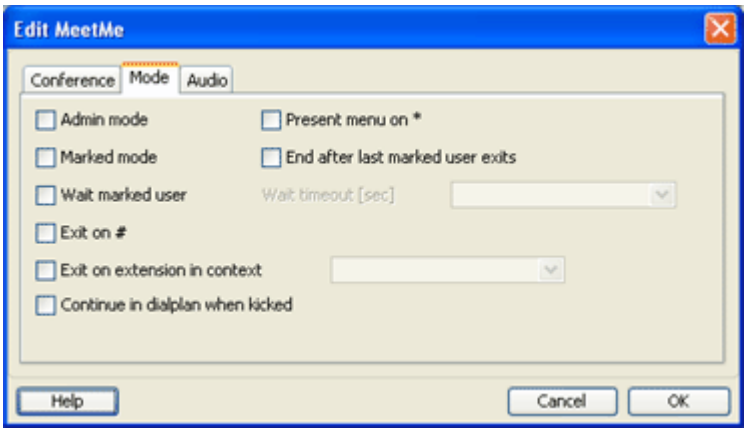
Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Field	Description
Conference	Conference room number. If the conference room number is omitted, the caller will be prompted to enter one.
PIN	Conference password i.e. Personal Identification Number (PIN).
Always prompt	Always prompt for the PIN number even if it is specified.
Dynamic conference	Create this conference on demand i.e. conference is created on the fly. User must input conference room number to create a dynamic conference. Dynamic conference room is created on the fly only if the specified conference room does not already exist. For example, if user A makes the dynamic conference #200, then user B comes along and also makes dynamic conference #200, it is considered correct behavior for user B to be joined to user A's conference instead of warning user B that conference #200 has already been created by someone. In addition, if the dynamic conference was created by user A to have a PIN, user B will instead be told 'invalid PIN' and then be prompted for a new conference number.

Field	Description
Dynamic conference with PIN	Conference on demand with a PIN. If you do not want a PIN assigned to the dynamic conference, just hit the '#' key when prompted for a PIN.
Select empty conf.	Select an empty conference room.
Select empty pinless conf.	Select an empty conference room and do not request PIN.
Run AGI Script	Execute specified AGI script. Default is conf-background.agi (Note: This does not work with non-Zap channels in the same conference).
Record conference in file	Record conference in a voice file. The default format of voice file is 'wav', and default filename is meetme-conf-rec-${CONFNO}-${UNIQUEID}.

Property Editor – Mode

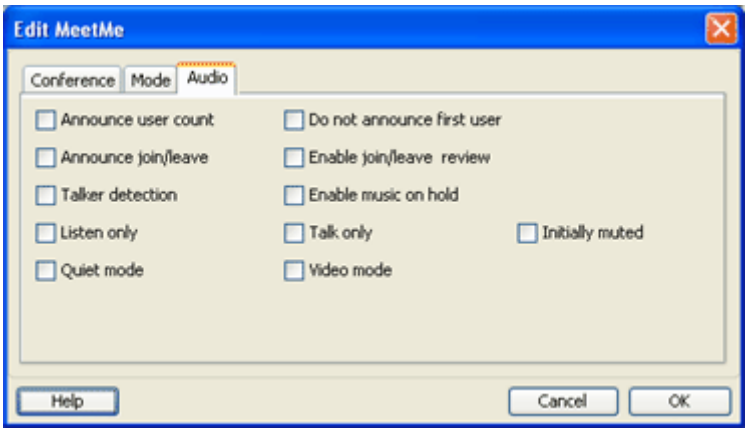


Field	Description
Admin mode	Set Admin mode for this conference.
Present menu on *	Present user with a menu: User (Admin mode should not be set): Upon pressing * plays the voicemenu "Please press 1 to mute or unmute yourself". Asterisk now comes with volume adjustment for the individual participants. By pressing * followed by 4 or 6 the user can adjust its volume. Admin (Admin mode should be set): Upon pressing * plays the voicemenu "Press 1 to mute/unmute yourself, 2 to lock/unlock this conference"
Marked mode	Set Marked mode for this conference.
End after last marked user exits	Close the conference when last marked user exits.
Wait marked user	Wait until the marked user enters the conference. All other connected users will hear MusicOnHold until the marked user enters.
Exit on #	Allow user to exit the conference by pressing '#' key.
Exit on extension in context	Allow user to exit the conference by entering a valid single digit extension of the specified context or the current context if the context is not specified.
Continue in dialplan when kicked	Allow caller to continue in the dialplan when kicked out of conference.

Note:

Neither the option 'Present menu on *' nor the option 'Always prompt' from Conference sheet will work if used together with option 'Exit on extension in context'.

Property Editor - Audio



Field	Description
Announce user count	Announce user(s) count on joining a conference.
Announce join/leave	Announce user(s) count on joining or leaving a conference.
Talker detection	Set talker detection (sent to manager interface and meetme list).
Listen only	Set monitor only mode (listen only, no talking).
Quiet mode	Quiet mode (don't play enter/leave sounds).
Do not announce first user	Do not play welcome message when first person enters the conferencing room.
Enable join/leave review	Announce user join/leave with review.
Enable music on hold	Enable music on hold when the conference has a single caller.
Talk only	Set talk only mode (talk only, no listening).
Initially muted	Set initially muted.
Video mode	This option currently does not have any function.

Note:

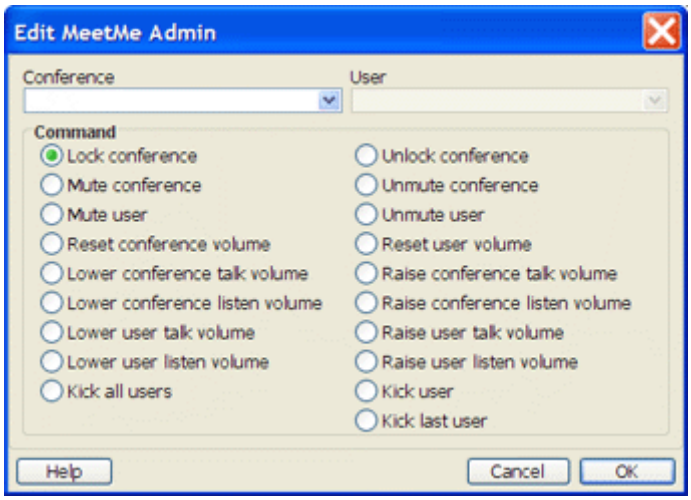
The MeetMe component requires a timer to work properly. There are several different ways to get the timer to work, but it won't work by default if you haven't got a Digium Zaptel hardware interface card installed. At this time only zaptel devices may be used. If you do not have a Zaptel device see the ztdummy instructions for timing.

MeetMeAdmin

MeetMe conference administration

This component could be used to run any of the available admin functions against conference participants or the conference.

Property Editor



Field	Description
Conference	MeetMe conference room number.
User	Conference user.
Lock conference	Lock the conference room i.e. do not allow new participants.
Mute conference	Mute all participants in this conference room.
Mute user	Mute selected user.
Reset conference volume	Reset all users volume settings.
Lower conference talk volume	Lower entire conference speaking volume.
Lower conference listen volume	Lower entire conference listening volume.
Lower user talk volume	Lower selected user's talk volume.
Lower user listen volume	Lower selected user's listen volume.
Kick all users	Kick all users from this conference room.
Unlock conference	Unlock the conference room i.e. allow new participants.
Unmute conference	Unmute all participants in this conference room.
Unmute user	Unmute user.
Reset user volume	Reset selected user's volume settings
Raise conference talk volume	Raise entire conference speaking volume.

Field	Description
Raise conference listen volume	Raise entire conference listening volume.
Raise user talk volume	Raise selected user's talk volume.
Raise user listen volume	Raise selected user's listen volume.
Kick user	Kick selected user (the user as defined in the 'User' field) from the conference.
Kick last user	Kick participants who last joined the conference.

Note:

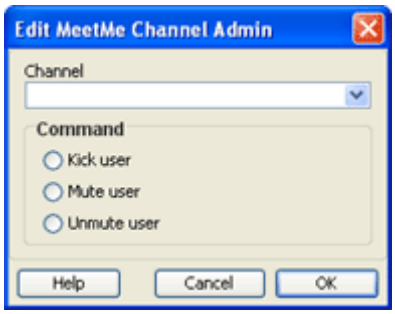
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

MeetMeChannelAdmin

MeetMe conference administration (channel specific)

This component could be used to run any of the available admin functions against conference participants or the conference.

Property Editor



Field	Description
Channel	MeetMe conference channel.
Kick user	Kick the specified user out of the conference he is in.
Mute user	Mute the specified user.
Unmute user	Unmute the specified user.

Note:

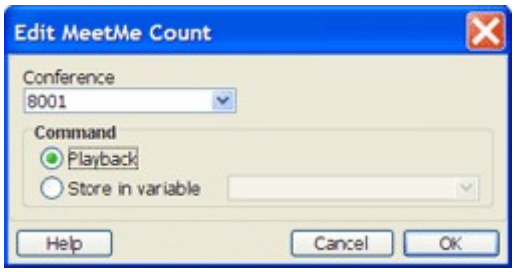
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

MeetMeCount

MeetMe participant count

This component plays back the number of users in the specified MeetMe conference.

Property Editor



Field	Description
Conference	MeetMe conference room.
Playback	Playback the announcement and say the number of participants in specified conference room.
Store in variable	If this options is selected, the playback will be skipped and the number of MeetMe participants will be stored in provided variable.

Note:

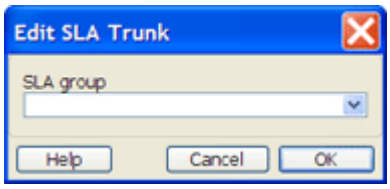
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

SlaTrunk

Run the Shared Line Appearance for a trunk

Runs the share line appearance for a trunk calling in. If there are no other participants in the conference, all member stations are invited into the bridge.

Property Editor



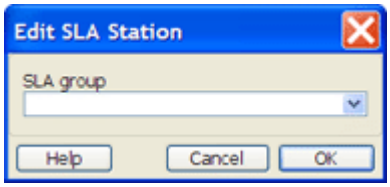
Field	Description
SLA group	SLA group.

SlaStation

Run the Shared Line Appearance for a station

Runs the share line appearance for a station calling in. If there are no other participants in the conference, the trunk is called and is dumped into the bridge.

Property Editor



Field	Description
SLA group	SLA group.

Queue category

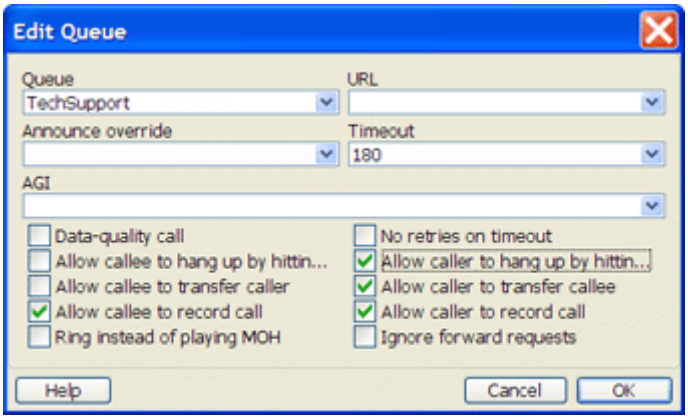
Queue category contains components responsible for queue and queue members management.

Queue

Queue a call

This component is used to queue an incoming call in a particular call queue as defined in queues.conf or as defined dynamically.

Property Editor



Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Field	Description
Queue	The name of the particular call queue as defined in queue.conf.
URL	A URL to be sent to the called party if the channel supports it.
Announce override	This option will override the announcement specified in queues.conf.
Timeout [seconds]	The time in seconds that a call will wait in the queue before routed further. The default is 300 seconds (5 minutes).

Field	Description
Data-quality call	Data-quality (modem) call (minimum delay).
Allow callee to hang up	Allow called user to hang up by pressing *.
Allow callee to transfer caller	Allow called user to transfer the caller.
Allow callee to record call	Allow called user to record the conversation.
Ring instead of playing MOH	Play ringing instead of music on hold (as defined in queue.conf).
No retries on timeout	Exit the component and continue with the call flow after the timeout.
Allow caller to hang up	Allow caller to hang up by pressing *.
Allow caller to transfer	Allow caller to transfer the call.
Allow caller to record	Allow caller to record the conversation.
Ignore forward requests	Ignore call forward requests from queue members and do nothing when they are requested.

Output Options

Output	Description
Timeout	The call was not answered during the specified timeout i.e. the caller timed out of the queue.
Full	The queue is full and the call can not be placed in the queue.
Join empty	The queue has no members.
Leave empty	The caller was in the queue, but all the members left the queue.
Join unavailable	The queue has no reachable members.
Leave unavailable	The caller was in the queue, but all the members became unreachable.

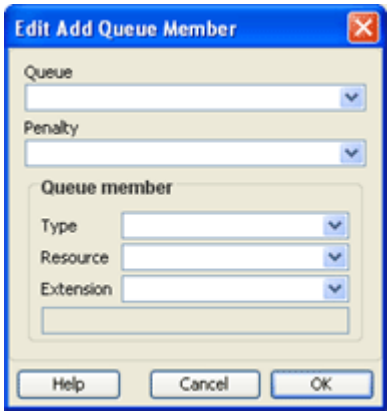
The application sets the following channel variable upon completion: QUEUESTATUS. The value could be one of the following: TIMEOUT, FULL, JOINEMPTY, LEAVEEMPTY, JOINUNAVAIL, or LEAVEUNAVAIL.

AddQueueMember

Dynamically add queue member

This component will dynamically add new member/agent into existing queue (no need to manually update queues.conf file). The entries in the agents.conf and queues.conf files will remain.

Property Editor



Field	Description
Queue	The name of the queue you want to add agent to.
Penalty	Using this option you can set a penalty to an queue member/agent. Penalty is similar to a agent priority. Agent with the lowest penalty will first receive a call in the queue.
Queue member	Target agent technology and resource (e.g. SIP/100).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

Output	Description
Success	Agent (technology/resource) is successfully added on the queue.
Already member	Agent is already a member of the queue.
No such queue	There is no such queue.

Example of Usage

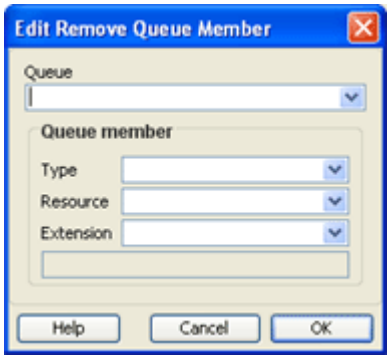
This component is usually used when a new agent is needed in the queue for a temporarily period of time.

RemoveQueueMember

Dynamically removes queue member

This component will dynamically remove existing member/agent from the queue (no need to manually update queues.conf file). The entries in the agents.conf and queues.conf files will remain.

Property Editor



Field	Description
Queue	The name of the queue to remove agent from.
Queue member	Target agent technology and resource (e.g. SIP/100).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

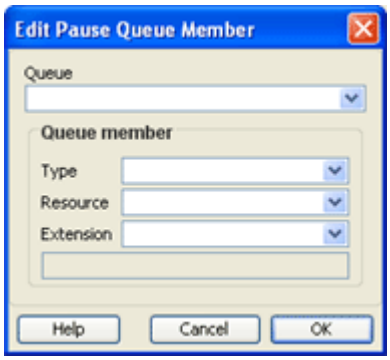
Output	Description
Success	Agent (technology/resource) is successfully removed from the queue.
Not in queue	The agent is not in the queue.
No such queue	There is no such queue.

PauseQueueMember

Pause a queue member

Pauses an agent on a queue, i.e. the agent will not receive calls but will remain the member of the queue. The entries in the agents.conf and queues.conf files will remain.

Property Editor



Field	Description
Queue	The name of the queue.
Queue member	Target agent technology and resource (e.g. SIP/100).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

Output	Description
Success	Agent (technology/resource) is successfully paused on the queue.
Not found	The agent is not in the queue.

Example of Usage

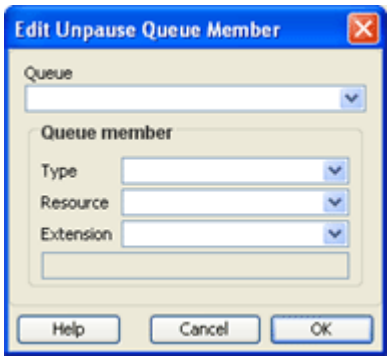
This component is usually used to temporary forbid an agent to receive calls in the queue. This restriction can be canceled using UnpauseQueueMember component.

UnpauseQueueMember

Unpause/resume a queue member

Unpause/resume an agent on a queue, i.e. the agent will continue to receive calls in the queue. The entries in the agents.conf and queues.conf files will remain.

Property Editor



Field	Description
Queue	The name of the queue.
Queue member	Target agent technology and resource (e.g. SIP/100).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Output Options

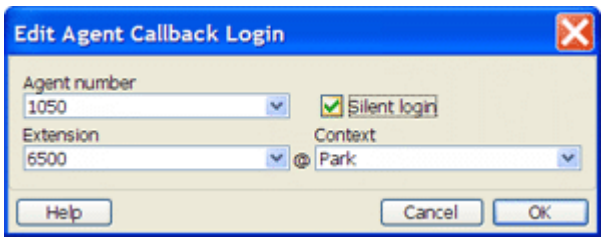
Output	Description
Success	Agent (technology/resource) is successfully unpaused/resumed on the queue.
Not found	The agent is not in the queue.

AgentCallbackLogin

Log-in an agent with callback

This component will log an agent into a queue and will allow callback. The agent's callback extension is called with the specified context.
The most important feature and the main difference between this component and the AgentLogin component is that with the AgentLogin you have to keep the phone receiver open. Otherwise, if you hang up the phone, the agent will be logged off from the queue. With the AgentCallbackLogin application, you are allowed to hang up the phone when the login process is finished and the user will not be removed from the queue.

Property Editor



Field	Description
Agent number	The agent you want to ask to log-in.
Silent login	The log-in announcement will not be played if this option is selected.
Extension	The extension to connect to an agent. Usually this is an extension with the Dial component. This component will connect the agent with an registered user. This field could be left blank and in that case the agent will be asked to enter extension during the login process.
Context	The context in the extensions.conf that contains the extension.

Note:

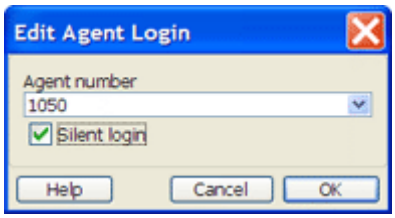
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

AgentLogin

Log-in an agent

This component will log in an agent. While logged in, the agent can receive calls and will hear a 'beep' when a new call arrives. The agent can dump the call by pressing the star key.

Property Editor



Field	Description
Agent number	The agent you want to ask to log-in.
Silent login	The log-in announcement will not be played if this option is selected.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Park

Park a call

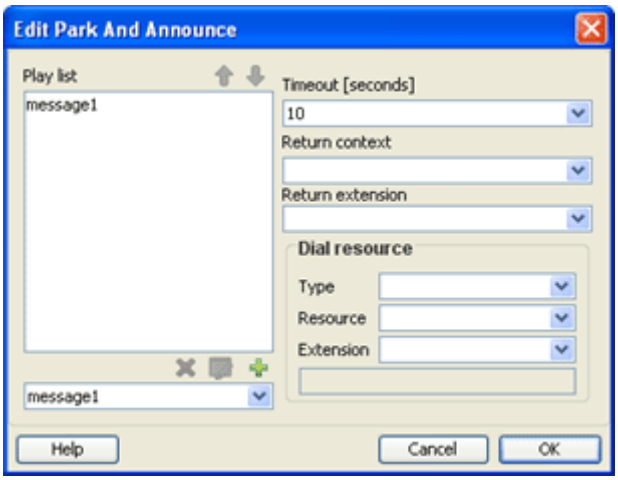
This component will park a call. The parkedcalls extension should be defined in order to use this component. The system will announce extension where the call is parked. By default the parked extension is 701.

ParkAndAnnounce

Park and announce call

This component will park a call and will dial another extension to announce this event.

Property Editor



Field	Description
Play list	A list of voice files to play or word PARKED. The word 'PARKED' will be replaced with the number of the extension the call is parked on and will be pronounced using Say component.
Timeout	Time in seconds before the call returns back to the return context.
Return context	The context to jump to after the timeout.
Return extension	The extension to jump to after the timeout.
Dial resource	Target device resource and extension (SIP/102, IAX/205 etc.)

Note:

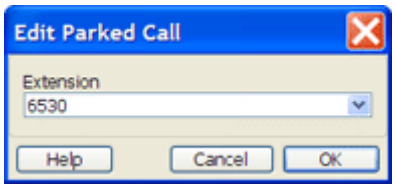
Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

ParkedCall

Answer a parked call

This component could be used to answer a parked call. In order to use this component the parkedcalls extension should be defined.

Property Editor



Field	Description
Extension	The extension number where is the parked call you want to answer.

Example of Usage

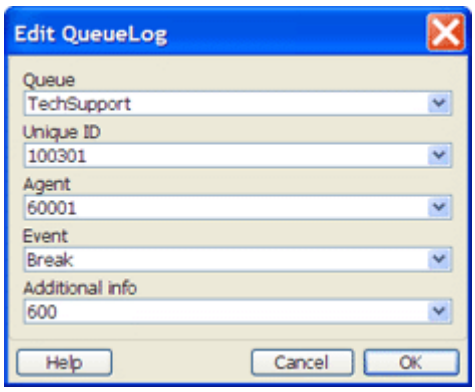
The parked call could be answered by dialing the parked extension or using the ParkedCall component.

QueueLog

Writes to the Queue log

This component is used to write your own events into the queue log.

Property Editor



Field	Description
Queue	Queue name.
Unique ID	Unique ID.
Agent	The agent.
Event	The event (your own event) you want to log into the queue.
Additional info	Additional info regarding your event.

Recording category

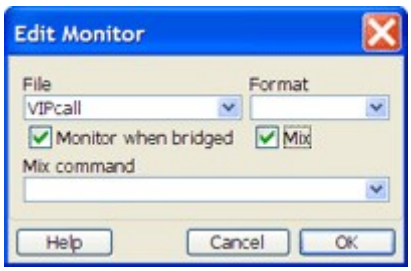
Recording category contains components responsible for channel recording.

Monitor

Record a telephone conversation to a sound file

The Monitor component will start recording a channel. The channel's input and output voice packets are saved to separate sound files. The filenames may be changed while recording using the ChangeMonitor component. Recording continues until either the StopMonitor component is reached or the channel hangs up.

Property Editor



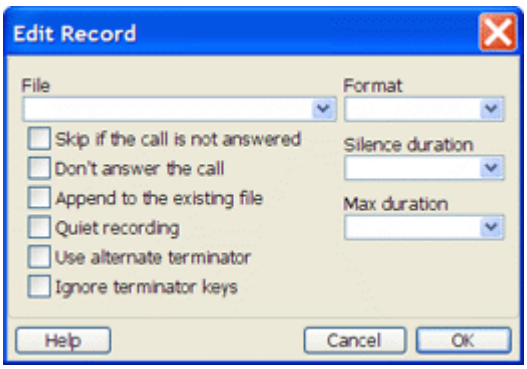
Field	Description
File	The base filename to use when saving the sound files. If not supplied, the default basename is constructed on the channel name plus a number, for example, IAX2[foo@bar]-3. The channel's input voice packets will be saved to basename-in.ext and the output voice packets will be saved to basename-out.ext.
Format	The sound file format to save in, which will be also used as the filename extension. The default is wav.
Monitor when bridged	If this option is selected Monitor will not begin recording unless a call is bridged.
Mix	If this option is selected, the Asterisk will execute a unix program to combine the two sound files into a single sound file when recording finishes. By default, Asterisk will execute soxmix and then delete the original two sound files. Note that sox/soxmix may not necessarily understand the sound format (e.g. alaw) and can't therefore mix the in and out files down to one single file. You may specify a different mixing method using Mix options to specify the path of the unix program you wish executed. At the completion of recording, the specified unix program will be executed.

Record

Record user voice input to a file

This component will record the current channel to a sound file saved with the given file name. The format specifies the sound format and the extension of the file. In case the full path to a voice file is not specified, the file will be stored in the /var/lib/asterisk/sounds folder. If the file with the same name and extension already exists, it will be overwritten. Recording stops when the specified Silence duration or Max duration is reached, when the '#' key is pressed, or when the channel is hung up.
Supported sound formats are: g723, g729, gsm, h263, ulaw, alaw, vox, wav, WAV (WAV is the GSM format of wav files).

Property Editor



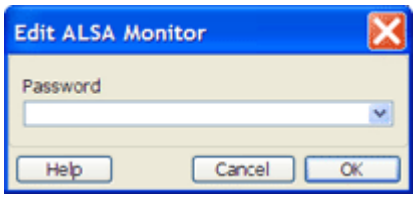
Field	Description
File	The base filename to use when saving the sound files.
Format	The sound file format to save in, which will be also used as the filename extension. Supported sound formats are: g723, g729, gsm, h263, ulaw, alaw, vox, wav, WAV (WAV is the GSM format of wav files).
Skip if call is not answered	Return immediately in case the call is not answered.
Don't answer the call	Record even if the call is not answered.

AlsaMonitor

Monitor the ALSA console

This component will allow user to monitor the ALSA console bi directionally. Monitoring is performed regardless of the state of any call that might be on the channel.

Property Editor



Field	Description
Password	Password.

MixMonitor

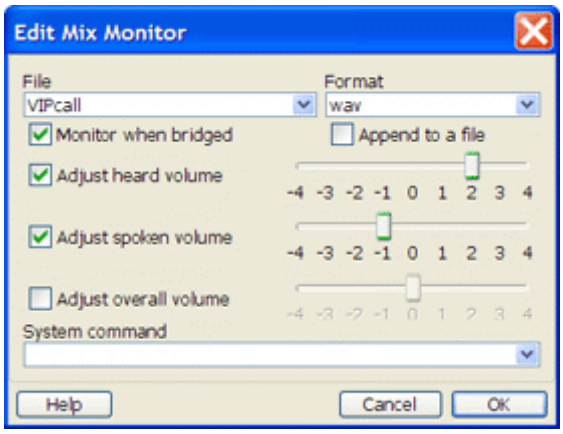
Record and mix call legs natively

This component is similar to the Monitor component with a difference in that MixMonitor is designed to record one audio file i.e. to mix both lags of the call natively while the call is in progress and that way avoid the need to call external processes which lead to harmful CPU usage spikes.

Benefits:

- One call can be recorded to multiple files at the same time
- Allows for recording a call to a single g729 file
- With 'Append to file' option and agent can record all his/her calls in one file
- The volume for either side of the channel may be adjusted separately

Property Editor



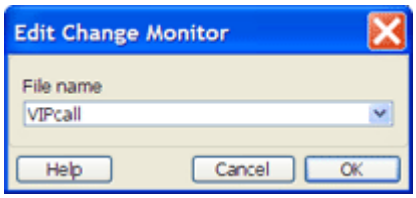
Field	Description
File	The base filename to use when saving the sound files.
Format	The sound file format to save in, which will be also used as the filename extension.
Monitor when bridged	Only save audio to the file while the channel is bridged. Please note that this *does not include conferences*.
Append to a file	Append new recordings to the file instead of overwriting it.
Adjust heard volume	Adjust the heard volume.
Adjust spoken volume	Adjust the spoken volume.
Adjust overall volume	Adjust the overall volume.
System command	Execute the system (Linux shell) command after the recording completion.

ChangeMonitor

Change monitoring filename of a channel

This component has effect only if the current channel is being monitored (being recorded using Monitor component). In that case, ChangeMonitor component could be used to change the filename of the file being saved to.

Property Editor



Field	Description
File name	The new filename to use when saving the sound files of currently being monitored channel.

StopMonitor

Stop monitoring a channel

This component stops monitoring a channel. The component has no effect if the channel is not monitored.

PauseMonitor

Pause monitoring of a channel

Pauses monitoring of a channel until it is re-enabled by a call to UnpauseMonitor.

UnpauseMonitor

Unpause monitoring of a channel

Unpauses monitoring of a channel on which monitoring had previously been paused with PauseMonitor.

StopMixMonitor

Stop recording a call through MixMonitor

This component stops the audio recording that was started with a MixMonitor component on the current channel.

Dictate

Records and plays back a dictation

This component will record and then playback dictation. The file would be recorded in raw format. The format could be changed using System command sox.

General options

dial 1 to toggle record and playback modes

dial 0 for help

dial * pause/unpause

dial # to enter a new filename

Playback Mode Options

dial 2 to toggle fast playback (speed 1x, 2x, 3x, 4x)

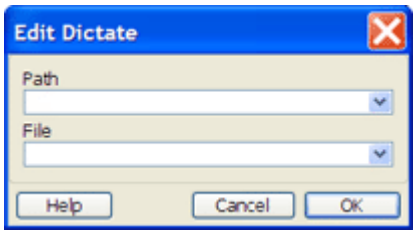
dial 7 to seek backwards a few frames

dial 8 to seek forwards a few frames

Record Mode Options

dial 8 to erase the whole file and start again.

Property Editor



Field	Description
Path	A full path to the folder to store the dictations to. The default is /var/spool/asterisk/dictate.
File	A file to store the dictations.

ChanSpy

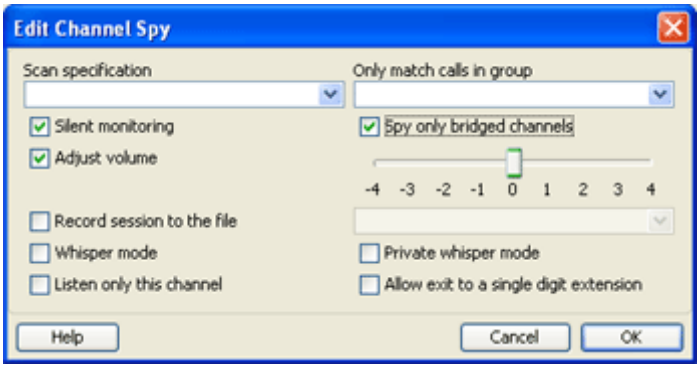
Universal channel barge-in i.e. listen to bridged call

This component could be used to listen to/spy on any bridged call, including VoIP only calls. This is especially useful in a call center deployments where supervisors monitor agents on the phone.

Use the following keys while listening to the channel:

- Dialing # cycles the volume level
- Dialing * will stop spying and look for another channel to spy on

Property Editor



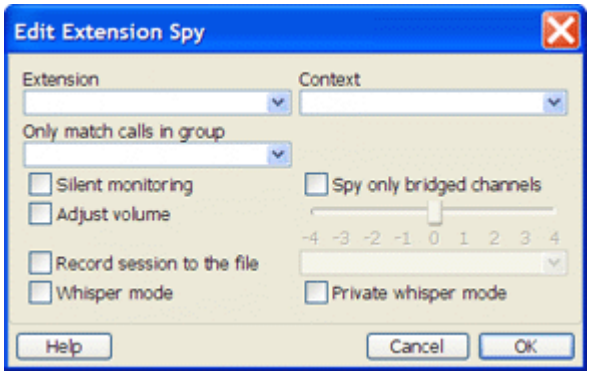
Field	Description
Scan specification	Scan only channels which titles *begins* with this string. Empty string is also valid.
Only match calls in group	Enforce group and only match calls in specified group.
Silent monitoring	Don't announce beep when recording a channel.
Spy only bridged channels	Only spy on channels involved in a bridged call.
Adjust volume	Adjust the initial volume (negative is quieter).
Record session to the file	Record the session to the file.
Whisper mode	Enable 'whisper' mode, so the spying channel can talk to the spied-on channel.
Private whisper mode	Enable 'private whisper' mode, so the spying channel can talk to the spied-on channel but cannot listen to that.
Listen only this channel	Listen only to audio coming from this channel.
Allow exit to a single digit extension	Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the SPY_EXIT_CONTEXT channel variable. The name of the last channel that was spied on will be stored in the SPY_CHANNEL variable.

ExtenSpy

Listen to a channel, and optionally whisper into it

This component is used to listen to the audio from an Asterisk channel. This includes both, the audio coming in and out of the channel.
While spying, the following actions may be performed:
- Dialing # cycles the volume level.
- Dialing * will stop spying and look for another channel to spy on.

Property Editor



Field	Description
Extension	Extension to spy.
Context	Context. If the optional context is not supplied, the current channel's context will be used.
Only match calls in group	Match only channels where their ${SPYGROUP} variable is set to contain 'grp' in an optional : delimited list.
Silent monitoring	Don't play a beep when beginning to spy on a channel.
Adjust volume	Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
Record session to the file	Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is 'chanspy'.
Whisper mode	Enable 'whisper' mode, so the spying channel can talk to the spied-on channel.
Spy only bridged channels	Only spy on channels involved in a bridged call.
Private whisper mode	Enable 'private whisper' mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.

Caller ID category

Caller ID category contains components responsible for Caller ID management like lookup caller ID name, set caller presentation etc.

LookupBlacklist

Look up Caller ID from blacklist database

This component looks up the Caller ID number on the active channel in the Asterisk database, in the family 'blacklist'. If the number is found, the call flow will branch and continue on the 'Block' port. Otherwise, the call flow will continue on the 'Allow' port.

Output Options

Output	Description
Allow	The Caller ID is not found in the blacklist database.
Block	The Caller ID is found in the blacklist database.

LookupCIDName

Look up Caller ID Name from local database

This component looks up for the Caller ID number on the active channel in the Asterisk database family 'cidname' and sets the Caller ID name. The component will do nothing if no Caller ID is received on the channel.

Example of Usage

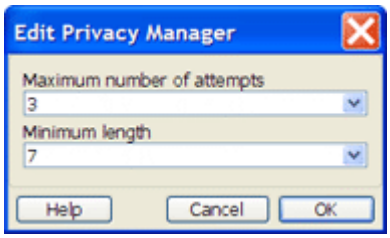
The component is usually used to change the name on some incoming call or to deliver Caller ID name if the incoming lines are not subscribed to the Caller ID name delivery.

PrivacyManager

Require phone number to be entered, if no Caller ID is sent

If no Caller ID is sent, the PrivacyManager component will answer the call and ask the caller to enter his/her phone number. The caller is given several attempts. If the caller failed after given number of attempt, the call flow will branch and continue on the Failed output port. In case the Caller ID is received on the channel, the call flow will be continued on the next component (Success output port).

Property Editor



Field	Description
Maximum number of attempts	Maximum number of attempts to enter phone number.
Minimum length	Minimum length of expected phone number (in digits).

Output Options

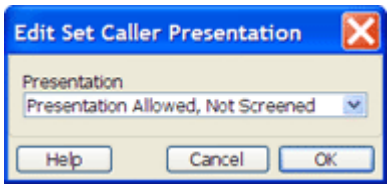
Output	Description
Success	The Caller ID is received on the channel or the caller successfully entered phone number.
Failed	Caller failed to enter his/her phone number within given number of attempts.

SetCallerPres

Set Caller ID presentation

This component will allow you to change the presentation of the Caller ID. The component should be used before placing an outgoing call.

Property Editor



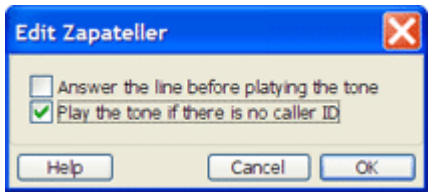
Field	Description
Presentation	Set Caller ID presentation to a new value. Available options are: Presentation Allowed, Not Screened Presentation Allowed, Passed Screen Presentation Allowed, Failed Screen Presentation Allowed, Network Number Presentation Prohibited, Not Screened Presentation Prohibited, Passed Screen Presentation Prohibited, Failed Screen Presentation Prohibited, Network Number Number Unavailable

Zapateller

Block telemarketers with SIT

This component generates a special information tone to block telemarketers.

Property Editor



Field	Description
Answer the line before playing the tone	If checked the line will be answered before playing the tone.
Play the tone if there is no caller ID	If checked the Zapateller component will play the tone only if there is no callerid information.

Billing category

Billing category contains components responsible for CDR records management.

NoCdr

Make sure Asterisk doesn't save CDR for the call

This component makes sure that there will be no any CDR records for the particular call.

ForkCdr

Fork the CDR into two separate entities

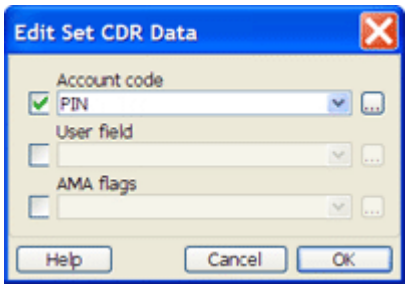
This component causes the CDR (Call Data Record) to fork an additional CDR record.

SetCdrData

Set some CDR fields

This component could be used to set some CDR fields. The following fields could be set with this component: Account code, User field, AMA flags.

Property Editor



Field	Description
Account code	Set Account code CDR field.
User field	Set User field CDR field.
AMA flags	Set AMA flags CDR field.

LumenVox category

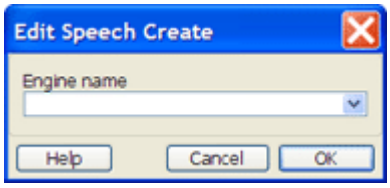
LumenVox category contains components responsible for speech recognition.

SpeechCreate

Create a Speech Structure

This component creates information to be used by all the other speech components. It must be called before doing any speech recognition activities such as activating a grammar. It takes the engine name to use as the argument. If not specified the default engine will be used.

Property Editor



Field	Description
Engine name	The speech recognition engine name.

SpeechDestroy

End speech recognition

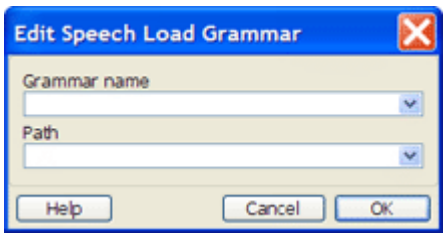
This component destroys the information used by all the other speech recognition applications and ends the speech recognition.
The component is located on the LumenVox sheet.

SpeechLoadGrammar

Load a Grammar

This component will load a grammar on the channel, not globally. The component takes the grammar name as first argument and path as second.

Property Editor



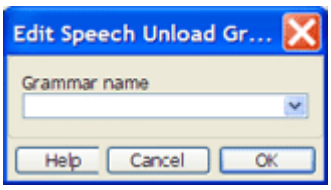
Field	Description
Grammar name	The grammar name.
Path	Full path to the file.

SpeechUnloadGrammar

Unload a Grammar

This component unloads the grammar. It takes the grammar name as the only argument.

Property Editor



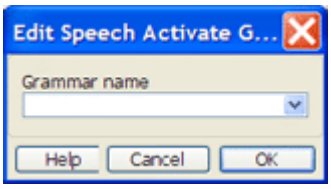
Field	Description
Grammar name	Grammar name.

SpeechActivateGrammar

Activate a Grammar

This component activates the specified grammar.

Property Editor



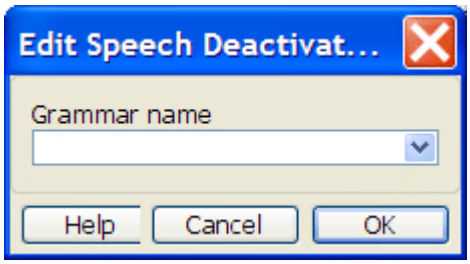
Field	Description
Grammar name	A grammar to loaded and activate.

SpeechDeactivateGrammar

Deactivate a Grammar

This component deactivates the specified grammar so that it is no longer recognized. The only argument is the grammar name.

Property Editor



Field	Description
Grammar name	Grammar to be deactivated.

SpeechStart

Start recognizing voice in the audio stream

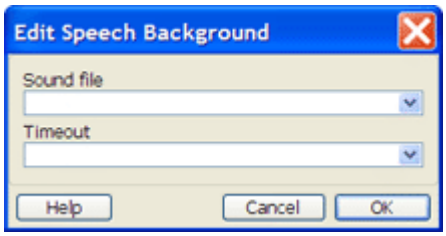
This component tells the speech recognition engine that it should start trying to get results from audio being fed to it.

SpeechBackground

Plays a sound file and waits for speech to be recognized

This component plays a sound file and waits for the caller to speak. Once the caller starts speaking the playback of the file stops. And then again, when the caller stops talking, the processing sound is played again to indicate that the speech recognition engine is working.
Once the results of the speech recognition are available the component returns and results (score and text) are available to be collected using other speech components or by reading related variables. The first text and score are stored in ${SPEECH_TEXT(0)} and ${SPEECH_SCORE(0)} while the second are stored in ${SPEECH_TEXT(1)} and ${SPEECH_SCORE(1)}.

Property Editor



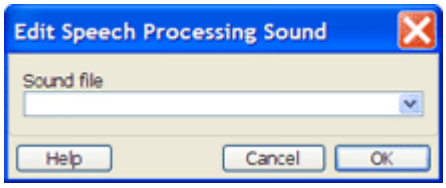
Field	Description
Sound file	A sound file to be played.
Timeout	Timeout. Note that timeout will start once the sound file has stopped.

SpeechProcessingSound

Change background processing sound

This component changes the processing sound that SpeechBackground plays back when the speech recognition engine is processing and working to get results. It takes the sound file as the only argument.

Property Editor



Field	Description
Sound file	The sound file.

Zap/Dahdi category

Zap/Dahdi category contains components responsible for Zap related actions.

Flash

Flashes a Dahdi Trunk

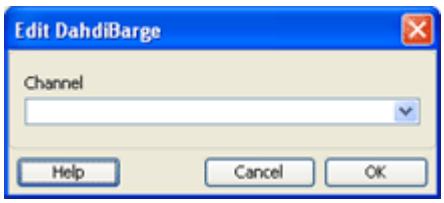
This component will send a flash on a Dahdi trunk. A switchook flash (or link) is an on-hook condition that lasts no less than 200ms, and no more than 1200ms. The actual length is different for different systems (400-600ms is probably the safest). You can perform a flash by pressing the link button on an analog set so equipped, or simply hold down the hookswitch for about half a second (be careful: too long and you'll hang up - too short and it'll be ignored). Note that in the UK BT chose to use a 80ms flash time (aka Recall). If you have any link-enabled features on your line, a flash is typically how you would access them.

ZapBarge

Listen to a Dahdi channel call

This component will enable you to listen_to/spy_on the conversation on a specified Dahdi channel, or will prompt if the channel is not specified. No indication is given to the other parties that their call is being listened to. This component could be used multiple times so several supervisors could listen to the same channel.

Property Editor



Field	Description
Channel	The channel to listen_to/spy_on. If the channel is not specified, the component will prompt the caller to enter the channel.

ZapScan

Scan Dahdi channels to monitor calls

This component allows you to monitor/listen_to Dahdi channels in a convenient way. Use '#' to select the next channel and '*' to exit.

ZapRas

Provide ISDN data service

The DahdiRAS component is for use with Dahdi channel ISDN connections. It provides a Remote Access Server (RAS) to allow data service on some of ISDN channels (internet connection for example). This component is not for use with analog lines; it does not provide a modem emulator.

Overview (for advanced users)

This component executes a RAS server using pppd on the given channel. Your pppd must be patched to be Dahdi-aware, also your kernel needs to be ppp-aware.

- Get pppd patched to support Zaptel/Dahdi. See ftp://ftp.digium.com/pub/zaptel/misc/
- You will need the tarball and both patches listed.
- Apply the PPPoE patch then the Zaptel/Dahdi patch.
- PPPoE support is required for kernel. Try to tag any ppp-related kernel options as modules.

At this point you should be able to compile and install the pppd daemon. After that you should re-compile Dahdi and then Asterisk it-self.

Property Editor

Field	Description
Value	A ISDN channel to execute a RAS server on.

ZapSendKeypadFacility

Send digits out of band over a PRI

This component will send the given string of digits in a Q.931 Keypad Facility IE out of band over the current Dahdi channel.

Property Editor

Field	Description
Digits	The string of digits to be sent over the current Zap channel.

SetTransferCapability

Set ISDN Transfer Capability

This component sets the ISDN transfer capability of a call to a new value.

Misc category

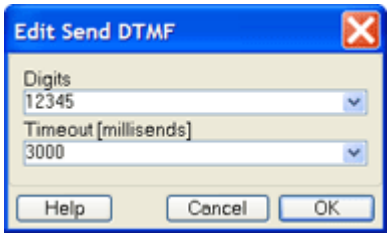
Misc category contains components responsible for various actions.

SendDtmf

Sends arbitrary DTMF digits

This component sends DTMF digits on a channel.

Property Editor



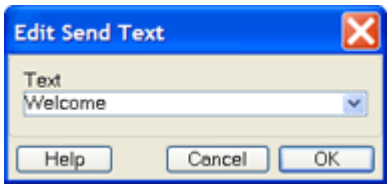
Field	Description
Digits	This field contains the value of DTMF digits to be sent on a channel. Accepted digits are 0-9 *#abcd.
Timeout	The time reserved for sending DTMF digits. All DTMF digits should be sent during this time.

SendText

Send text to the client on the connected channel

This component accepts a text string as argument and attempts to send it to the calling client via the sendtext function of the channel driver. The component does not encode characters in any special way, it simply passes the given text buffer to the channel driver service routine.

Property Editor



Field	Description
Text	Text to be sent to the client on the connected channel.

Output Options

Output	Description
Success	The text message is sent successfully on the connected channel.
Failure	The text message could not be sent.
Unsupported	The channel does not support text messaging.

Example of Usage

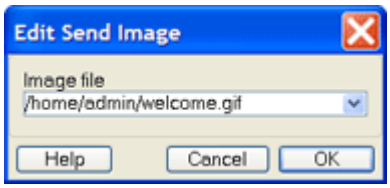
This component could be used for advertising purposes to send a text message when someone attempts to call the company. You could also use this function to send text message to a calling party if you are busy or unavailable.

SendImage

Sends image file on a channel

This component accepts image file as argument and attempts to send it to the calling client.

Property Editor



Field	Description
Image file	A full path to the image file to be sent to the client on the connected channel.

Output Options

Output	Description
Success	The image is sent successfully on the connected channel.
Unsupported	The channel does not support images.

Example of Usage

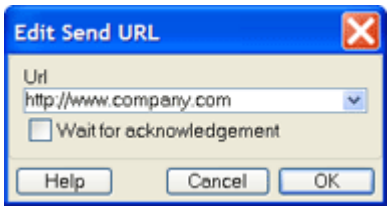
This component could be used to display a company logo (or any other) picture on the screen of the caller when the company's phone number is dialed and the channel established.

SendURL

Send URL on a channel

This component sends URL address to the caller and requires caller's softphone/phone to open the URL.

Property Editor



Field	Description
Url	This field contains the URL to be sent to the client on the connected channel.
Wait for acknowledgement	Wait for the acknowledgement that the URL has been loaded before continuing with the dial plan flow.

Output Options

Output	Description
Success	URL successfully sent to client. Channel variable SENDURLSTATUS is set to SUCCESS value.
Failure	Failed to send URL. Channel variable SENDURLSTATUS is set to FAILURE value.
Unsupported	Channel does not support URL transport. Channel variable SENDURLSTATUS is set to UNSUPPORTED value.

Example of Usage

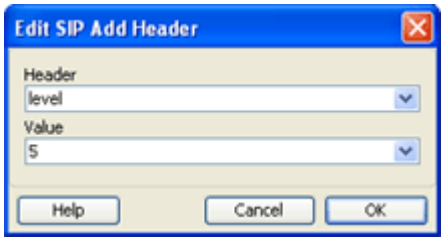
This component could be used for advertising purposes, for example, to send a URL of company's web site when the caller dials the company's phone number.

SipAddHeader

Add SIP header to outgoing call

This component adds the key/value pair as a SIP header to outgoing call. The component is located on the Misc sheet.

Property Editor



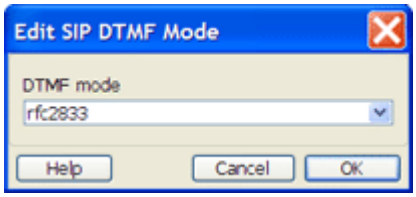
Field	Description
Header	Variable name (key).
Value	The value assigned to the variable.

SipDtmfMode

Change the DTMF mode for a SIP call

This component changes the DTMF mode for a SIP call. The component influence only the calls originating in a SIP channel, not the calls TO a SIP channel originating in another type of channel, like a ZAP channel.

Property Editor



Field	Description
DTMF mode	Select type of DTMF mode for a SIP call.

Iax2Provision

Provision a calling IAXy with a given template

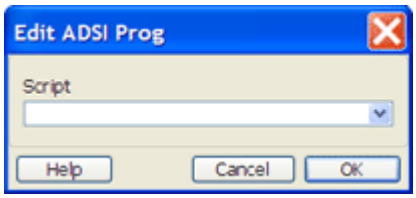
This component provisions the calling IAXy (assuming the calling entity is in fact an IAXy) with the given template or default if one is not specified. Returns -1 on error or 0 on success.

AdsiProg

Load Asterisk ADSI script into the phone

This component will allow you to programs an ADSI Phone with the given script.

Property Editor



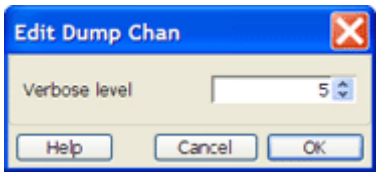
Field	Description
Script	The script parameter can be an absolute file path, relative file path, or can be left empty. If it is relative, the root of the path is the Asterisk config directory i.e. AdsiProg component would attempt to use the file from /etc/asterisk folder.

DumpChan

Dump info about the calling channel

This component will dump the information about the calling channel, including the information about the channel and a complete list of all the channel variables. These information could be seen on the Asterisk console. This application is usually used for debugging purposes.

Property Editor



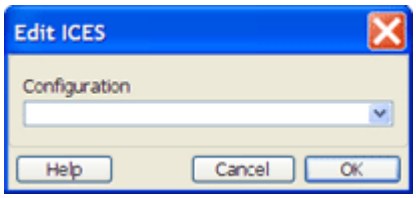
Field	Description
Verbose level	A minimum level of verbosity.

Ices

Stream to an icecast server

This component will allow you to stream an Asterisk channel into the internet.

Property Editor



Field	Description
Configuration	A full path to the Ices configuration xml file.

NbsCat

Play an NBS local stream

This component executes nbscat to listen to the local NBS stream. User can exit by pressing any key.

TestServer

Execute interface Test Server

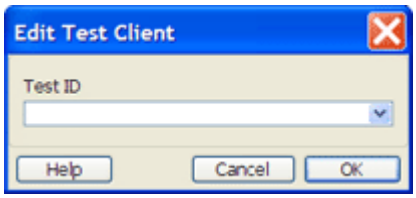
This component performs test server function and writes call report. Results are stored in the /var/log/asterisk/testreports/<testid>-server.txt file, or /var/log/asterisk/testresults/<testid>-server.txt file depending on the Asterisk version. This component should be used with TestClient component.

TestClient

Execute interface Test Client

This component executes test client with given testid. Results are stored in /var/log/asterisk/testreports/<testid>-client.txt file. This component should be used with TestServer component.

Property Editor



Field	Description
Test ID	The test ID.

SMS

Exchange SMS data

This component handles calls to/from text message capable phones and message centers using ETSI ES 201 912 protocol 1 FSK messaging over analogue calls. Basically it allows sending and receiving of text messages over the PSTN. It is compatible with BT Text service in the UK and works on ISDN and PSTN lines. It is designed to connect to an ISDN or zap interface directly and uses FSK so would probably not work over any sort of compressed link. Landline SMS is starting to be available in various parts of Europe, and is available from BT in the UK. However, Asterisk would allow gateways to be created in other locations such as the US, and use of SMS capable phones such as the Magic Messenger.

Background

Short Message Service (SMS), or texting is very popular between mobile phones. A message can be sent between two phones, and normally contains 160 characters. There are ways in which various types of data can be encoded in a text message such as ring tones, and small graphic, etc. Text messaging is being used for voting, competitions...

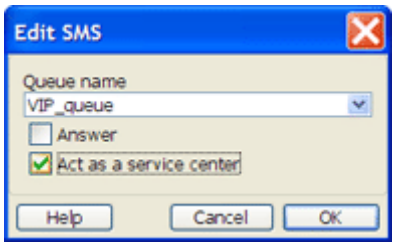
Sending a message involves the mobile phone contacting a message center (SMSC) and passing the message to it. The message center then contacts the destination mobile to deliver the message. The SMSC is responsible for storing the message and trying to send it until the destination mobile is available, or a timeout.

Landline SMS works in basically the same way. You would normally have a suitable text capable landline phone, or a separate texting box such as a Magic Messenger on your phone line. This sends a message to a message center your telco provides by making a normal call and sending the data using 1200 Baud FSK signaling according to the ETSI spec. To receive a message the message center calls the line with a specific calling number, and the text capable phone answers the call and receives the data using 1200 Baud FSK signaling. This works particularly well in the UK as the calling line identity is sent before the first ring, so no phones in the house would ring when a message arrives.

Terminology

Field	Description
SMS	Short Message Service, i.e. text messages.
SMSC	Short Message Service Center. The system responsible for storing and forwarding messages.
MO	Mobile Originated. A message on its way from a mobile or landline device to the SMSC.
MT	Mobile Terminated. A message on its way from the SMSC to the mobile or landline device.
RX	Receive. A message coming in to the Asterisk box.
TX	Transmit. A message going out of the Asterisk box.

Property Editor



Field	Description
Queue name	By default all queues are held in a director /var/spool/asterisk/sms. Within this directory are sub directories mtrx, mttx, morx, motx which hold the received messages and the messages ready to send. Also, /var/log/asterisk/sms is a log file of all messages handled.
Answer	Send initial FSK packet.
Act as a service center	Act as a service centre talking to a phone.

Example of Usage

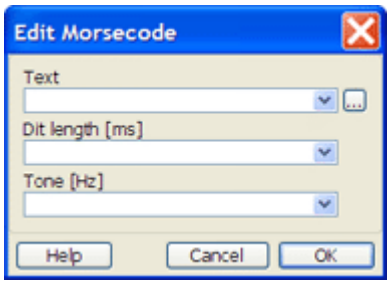
Sending messages from an Asterisk box can be used for a variety of reasons, including notification from any monitoring systems, email subject lines, etc.
Receiving messages to an Asterisk box is typically used just to email the messages to someone appropriate - we email and texts that are received to our direct numbers to the appropriate person. Received messages could also be used to control applications, manage competitions, votes, post items to IRC etc.

Morsecode

Plays Morse code

Plays the Morse code equivalent of the passed string.

Property Editor



Field	Description
Text	The text to be played using Morse code.
Dit length	Length of the dit. The default is 80ms.
Tone	The default tone is 800Hz.

Chapter

5

Visual Dialplan Functions

Visual Dialplan functions are accessible through one component, the Set component () available at the Variable component category.

The functions are grouped under the following categories:

- ACD
- Call
- Data
- Math
- Logical
- Other
- SIP
- Speech
- String

In this chapter we will describe in more detail each function.

ACD functions group

ACD functions group contains functions responsible for ACD related actions like get more info about an agent, get number of members in the queue etc.

Agent

Gets information about an Agent

This function could be used to get more information about an agent.

Property Editor

Field	Description
Agent ID	The ID of the Agent you want to get more info about.
Type	The valid items to retrieve are: Status. Status of the agent (Logged In or Logged Out). Password. The password of the agent. Name. The name of the agent. Hold music class. Music on hold class. Call back extension. The callback extension for the Agent (AgentCallbackLogin). Active channel. The name of the active channel for the Agent (AgentLogin).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

QueueMember

Count number of members answering a queue

This function returns the number of members currently associated with the specified queue.

Property Editor

Field	Description
Queue	Queue name.
Type	One of three options may be passed to determine the count returned: - Number of logged-in members for the specified queue - Number of logged-in members for the specified queue available to take a call - Number of all members for the specified queue

QueueMemberCount

Count number of members answering a queue

This function returns the number of members currently associated with the specified queue.

Property Editor

Field	Description
Queue	Queue name.

QueueMemberList

Returns a list of interfaces on a queue

This function returns a comma-separated list of members associated with the specified queue.

Property Editor

Field	Description
Queue	Queue name.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

QueueWaitingCount

Count number of calls currently waiting in a queue

This function returns the number of callers currently waiting in the specified queue.

Property Editor

Field	Description
Queue	Queue name.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Call functions group

Call functions group contains functions responsible setting and management call related information like CallerID, Timeout etc.

CallerId

Get or set Caller ID

This function could be used to get or set Caller ID value.

Property Editor

Field	Description
Type	Available values are: * Name: alphanumeric string * Number: number at which caller prefers to be called back (digits only) * All: a Caller ID string with the number specified between angle-brackets, e.g. "Some User <123>" * ANI: billing number (digits only) * DNIS: (digits only) * RDNIS: (digits only)

Timeout

Gets or sets timeouts on the channel

This function may be used to get or set various channel timeouts.

Property Editor

Field	Description
Type	The timeouts that can be manipulated are: **Absolute**: The absolute maximum amount of time permitted for a call. A setting of 0 disables the timeout. **Digit**: The maximum amount of time permitted between digits when the user is typing in an extension. When this timeout expires, after the user has started to type in an extension, the extension will be considered complete, and will be interpreted. Note that if an extension typed in is valid, it will not have to timeout to be tested, so typically at the expiry of this timeout, the extension will be considered invalid (and thus control would be passed to the Invalid extension, or if it doesn't exist the call would be terminated). The default timeout is 5 seconds. **Response**: The maximum amount of time permitted to type an extension. If the user does not type an extension in this amount of time, control will pass to the Timeout extension if it exists, and if not the call would be terminated. The default timeout is 10 seconds.

GroupList

Returns a list of all the groups set on a channel

This function will return a space separated list of all the groups set on a channel.

Group

Gets or sets the channel group

This function could be used to get or set the channel group.
Group is somewhat like a global variable repository. You give it names and it takes care of variable globally incrementing and decrementing the value of that name for each instance of the dial plan that is using a channel. For example, if five users make outgoing calls within the same group name assigned to each individual dial plan instance then the GroupCount function for that group name will return 5.

Property Editor

Field	Description
Category	The category name.

GroupCount

Counts the number of channels in the specified group

This function calculates the number of channels in the specified group, or uses the channel's current group if not specified.

Property Editor

Field	Description
Group	The group of channels.
Category	The category.

GroupMatchCount

Counts the number of channels in the groups matching the specified pattern

This function will calculate the Group Count for all groups that match the specified pattern. The function supports standard regular expressions.

Property Editor

Field	Description
Group pattern	Regular expression.
Category	A category of group.

Channel

Gets/set various pieces of information about the channel

Item may be one of the following:

Read/write	Item	Description
R/O	audioreadformat	format currently being read
R/O	audionativeformat	format used natively for audio
R/O	audiowriteformat	format currently being written
R/W	callgroup	call groups for call pickup
R/O	channeltype	technology used for channel
R/W	language	language for sounds played
R/W	musicclass	class (from musiconhold.conf) for hold music
R/O	state	state for channel
R/O	tonezone	zone for indications played
R/O	videonativeformat	format used natively for video

Notes:
Those items marked R/W above may be both read and set.

- Additional items may be available from the channel driver providing the channel; see its documentation for details.
* Any item requested that is not available on the current channel will return an empty string.

Blacklist

Check if the callerid is on the blacklist

This function uses astdb to check if the Caller*ID is in family 'blacklist'. Returns 1 or 0.

Lock

Attempt to obtain a named mutex

This functions attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock.

Returns 1 if the lock was obtained or 0 on error.

Note:

To avoid the possibility of a deadlock, LOCK will only attempt to obtain the lock for 3 seconds if the channel already has another lock.

TryLock

Attempt to obtain a named mutex

This functions attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock. Returns 1 if the lock was available or 0 otherwise.

Unlock

Unlocks a named mutex

Unlocks a previously locked mutex.
Note that it is generally unnecessary to unlock in a hangup routine, as any locks held are automatically freed when the channel is destroyed. Returns 1 if the channel had a lock or 0 otherwise.

Cdr

Get or set CDR data

This function could be used to get or set CDR data.

Property Editor

Field	Description
Type	Type may be one of the following: * Caller ID * Source * Destination * Destination context * Channel name * Destination channel * Last app executed * Last app's arguments * Start time - Time the call started * Answer time - Time the call was answered * End time - Time the call ended * Duration - Duration of the call * Duration after answer - Duration of the call once it was answered * Disposition - Answer, No answer, Busy * AMA flags - Documentation, Bill, Ignore etc. * Account code - The channel's account code * Channel unique id - The channel's unique id * User field - The channel's user specified field or any custom value that you wish to store
Search entire stack of CDRs	This option will cause the function to search the entire stack of CDRs on the channel for the requested value.
Return raw values	This option will cause the function to retrieves the raw, unprocessed value. For example, 'Start time', 'Answer time', and 'End time' will be retrieved as epoch values, when this option is checked, or formatted as YYYY-MM-DD HH:MM:SS otherwise. Similarly, disposition and AMA flags will return their raw integral values. Raw values for Disposition: 1 = NO ANSWER 2 = BUSY 3 = FAILED 4 = ANSWERED Raw values for AMA flags: 1 = OMIT 2 = BILLING 3 = DOCUMENTATION

Data functions group

The Data functions group contains functions responsible setting and management of database records and variables.

Db

Read or Write from/to the Asterisk database

This function provides inline access to the values in the Asterisk Database without executing a separate component. The function may read and written to the database. On a read, this function returns the corresponding value from the database, or blank if it does not exist (however, use DB_EXISTS function if you want to check if an database entry exists). Reading a database value will also set the variable DB_RESULT.

Property Editor

Field	Description
Family	Database family name.
Key	Database key name.

DbExists

Check to see if a key exists in the Asterisk database

This function will check to see if a key exists in the Asterisk database. Checking for existence of a database key will also set the variable DB_RESULT to the key's value if it exists.

Property Editor

Field	Description
Family	Database family name.
Key	Database key name.

Sort

Sorts a list of keys and values

This function takes a list of keys and values and returns a comma-delimited list of the keys, sorted by each associated value. Values are sorted as floats.

Property Editor

Field	Description
Key	Key name.
Value	Key value.

Example of Usage

For example if the list of key:value pairs is as the following:
three:3
two:2
one:1
hundred:100
negone:-1

Then the result of this function would be:
negone, one, two, three, hundred

Env

Gets or sets specified environment variable

This function reads or sets the specified environment variable.

Property Editor

Field	Description
Variable	A variable to read from or write to.

Eval

Evaluates stored variables

This function basically causes a variable/expression to be evaluated twice. When a variable or expression is in the dialplan, it will be evaluated at runtime. However, if the result of the evaluation is in fact a variable or expression, this function will make sure it is evaluated a second time. For example, if the variable ${MYVAR} contains "${OTHERVAR}", then the result of putting ${EVAL(${MYVAR})} in the dialplan will be the contents of the variable, OTHERVAR. Normally, by just putting ${MYVAR} in the dialplan, you would be left with "${OTHERVAR}".

Property Editor

Field	Description
Variable	A variable to be evaluated.

Set

Assigns a value to a channel variable

This function will store a value to a channel variable. The function is usually used to store the value of a complex expression to a channel variable, when calculating even more complex expressions.

Property Editor

Field	Description
Variable	A variable to store the value to.
Value	A value to store to the variable.

DbDelete

Return a value from the database and delete it

This function will retrieve a value from the Asterisk database and then remove that key from the database. DB_RESULT variable will be set to the key's value if it exists.

Property Editor

Field	Description
Family	Database family.
Key	The key for selected database family.

Global

Gets or sets the global variable specified

This function will get or set a global variable.

Property Editor

Field	Description
Variable	A global variable to be set or get.

Math functions group

Math functions group contains functions responsible for mathematical operations like Rand, Sha1 etc.

Math

Perform floating point calculation

This function performs mathematical calculation.

Md5

Computes an MD5 digest

This function computes an MD5 digest.

RandFunction

Choose a random number in a range

This function choose a random number between min and max. Min defaults to 0, if not specified, while max defaults to RAND_MAX i.e. 2147483647 on most of systems.

Sha1

Computes a SHA1 digest

This function generate a SHA1 digest via the SHA1 algorithm. Returns the 40 hex character digest string.

Logical functions group

Logical functions group contains functions that evaluates some logical expressions and based on the results returns true or false (1 or 0) like If, IfTime etc.

If

Returns one of the results based on the condition

This function will evaluate the condition and based on the result return the true or false result.

Property Editor

Field	Description
Condition	A condition that will be evaluated in order to decide which result should be returned.
True result	A result that should be returned in case the true condition.
False result	A result that should be returned in case the false condition.

IfTime

Returns one of the results based on the current time

This function will return the true or false result based on the current time.

Property Editor

Field	Description
Time condition	A condition that will be evaluated in order to decide which result should be returned.
True result	A result that should be returned in case the true condition i.e. current time is in the defined time frame.
False result	A result that should be returned in case the false condition i.e. current time is not in the defined time frame.

IsNull

Tests whether a value is NULL

This function checks if a variable value is NULL and returns 1 in case the value is NULL or 0 otherwise.

Property Editor

Field	Description
Variable	A variable that will be checked.

Exists

Tests whether string is empty

This function checks the existence of a string and returns 1 if the string is not empty i.e. exists or 0 otherwise.

Property Editor

Field	Description
Variable	A variable that should be checked.

CheckMd5

Checks an MD5 digest

This function checks MD5 digest and returns 1 on a match, 0 otherwise.

Property Editor

Field	Description
Digest	A digest that should be checked.
Data	A data to use when checking.

Other functions group

The Other functions group contains various functions like DundyLookup, VoiceMailCount etc.

IaxPeer

Get IAX peer information

This function will return IAX peer information.

Property Editor

Field	Description
Current channel	If this option is checked, the function will return IP address of the current channel
Peer	Peer name. If this option is checked, the items under the Info drop-down box are valid.
Info	Valid items only when the Peer option is selected. The items are: * IP address * Status - the peer's status (if qualify=yes) * Mailbox - the configured mailbox * Context - the configured context * Expire time - the epoch time of the next expire * Dynamic status - is it dynamic? (yes/no) * Caller ID name - the configured Caller ID name * Caller ID number - the configured Caller ID number * Codecs - the configured codecs * Preferred codec

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

DundiLookup

Performs DUNDi lookup of a phone number

This function will do a DUNDi lookup of the given phone number. If no context is given, the default will be e164. The result of this function will be the Technology/Resource found in the DUNDi lookup. If no results were found, the result will be blank.

Property Editor

Field	Description
Number	A number to use when performing DUNDi lookup.
Context	A context to use for a search.
Bypass internal DUNDi cache	If checked the internal DUNDi cache will be excluded from the search.

EnumLookup

Query NAPTR records

This function could be used for general or specific query against NAPTR records or counts of NAPTR types for ENUM or ENUM-like DNS pointers.

Property Editor

Field	Description
Number	Enum lookup number.
Method	A method to use when performing Enum lookup. Method 'ALL' will perform lookup against all NAPTRs. The default method is 'sip'.
Count records	If checked the EnumLookup will return an integer count of the number of NAPTRs of a certain RR type.
Record index	The default record index is '1'.
Prefix	The default prefix is 'e164.arpa'.

TxtCidName

Looks up a caller name via DNS

This function looks up the given phone number in DNS to retrieve the caller id name. The result would be either the blank or the value found in the TXT record in DNS.

Property Editor

Field	Description
Number	A number to use when searching for a caller name.

QueueAgentCount

Counts number of queue members

This function will return number of queue members (agents who are answering calls placed in the queue) for selected queue.
The function is located under the Other group.

Property Editor

Field	Description
Queue	Queue name.

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. You can define a connection to the Asterisk server in the Preferences dialog.

VmCount

Counts the voicemail messages in the specified voicemail box

This function counts the number of voicemail messages in the specified voicemail box.

Property Editor

Field	Description
Mailbox	A voicemail box to use when counting the voicemail messages.
Context	A context where the voicemail box exists.
Folder	Specific folder of the voicemail box (Inbox, Business etc.).

Note:

Resources listed in blue color are resources read from the Asterisk server configuration files. Make sure to define the connection to the target Asterisk server in order to enable Visual Dialplan to read the configuration data and to simplify your dialplan development. You can define a connection to the Asterisk server in the Preferences dialog.

Curl

Retrieves the content of a URL

This function will retrieve a content for given URL.

Property Editor

Field	Description
GET/POST method	A http method that should be used.
URL	URL to retrieve a content from.
Parameter name/value	Parameters that should be passed with the URL.

Stat

Does a check on the specified file

This function will check a specified file.

Property Editor

Field	Description
File	Full path to the file.
Type	You may check if the file type is one of the following: - Checks if the file is a directory - Checks if the file exists - Checks if the file is a regular file - Return the file mode (in octal) - Return the size (in bytes) of the file - Return the epoch at which the file was last accessed - Return the epoch at which the inode was last changed - Return the epoch at which the file was last modified

Sysinfo

Returns system information specified by parameter

This function returns information from a given parameter.

Sip functions group

The Sip functions group contains various functions responsible for getting and setting Sip related info.

SipChanInfo

Gets the specified SIP parameter from the current channel

This function can be used to get a number of SIP parameters from the current channel.

Property Editor

Field	Description
Info type	Available options are: * IP address of the peer * source IP address of the peer * URI from the From: header * URI from the Contact: header * Useragent * Name of the peer

SipPeer

Get SIP peer information

This function will get a SIP peer information.

Property Editor

Field	Description
Peer	A peer to get SIP information about.
Info	Valid options are: * IP address * Port number * Mailbox - the configured mailbox * Context - the configured context * Expire time - the epoch time of the next expire * Dynamic status - is it dynamic? (yes/no) * Caller ID Name - the configured Caller ID name * Caller ID Number - the configured Caller ID number * Call group - the configured Call group * Pickup group - the configured Pickup group * Codecs - the configured codecs * Status - status (if qualify=yes) * Registration extension * Call limit * Current number of calls. Only available if call-limit is set * Default language for peer * Account code - value accountcode field of CDR records for calls coming from this peer * User agent - current user agent id for peer * Preferred codec

CheckSipDomain

Checks if domain is a local domain

This function checks if the domain in the argument is configured as a local SIP domain that this Asterisk server is configured to handle. Returns the domain name if it is locally handled, otherwise an empty string.

Property Editor

Field	Description
Domain	A domain to be checked.

SipHeader

Gets the specified SIP header

This function can be used to set or get a sip header.

Property Editor

Field	Description
Header	A SIP header that should be set or read.
Index	Since there are several headers (such as Via) which can occur multiple times, this argument should be used to specify which header with the name specified under 'Header' field should be retrieved. Headers start at offset 1.

Speech functions group

The Speech functions group contains various functions related to the speech engine.

Speech

Gets information about speech recognition results

This function returns information about a speech recognition result.

Property Editor

Field	Description
Type	Available types are: Existence status - Returns 1 if speech object exists, or 0 otherwise. Spoke status - Returns 1 if caller spoke, or 0 otherwise. Number of results - Returns number of results that were recognized.

SpeechEngine

Change a speech engine specific attribute

This function can change a speech engine specific attribute.

Property Editor

Field	Description
Attribute	Specific speech engine attribute you want to get changed.

SpeechGrammar

Gets the matched grammar of a result if available.

This function gets the matched grammar of a result if available.

Property Editor

Field	Description
Result index	Matched grammar, if available.

SpeechScore

Gets the confidence score of a result

This function will return the confidence score of a result.

Property Editor

Field	Description
Result index	Resulting score index.

SpeechText

Gets the recognized text

This function will get the recognized text of a result.

Property Editor

Field	Description
Result index	Recognized text.

String functions group

String functions group contains various functions responsible for string manipulation.

Len

Returns the length of the given text

This function returns integer length of the string.

Property Editor

Field	Description
Text	A string.

Cut

String parsing based upon a delimiter

This function could be used for advanced string parsing based on a delimiter. The function is most useful when dealing with fields that are variable width. For fixed width string processing, use the built-in variable substitution.

Property Editor

Field	Description
Variable	A variable you want to cut. This parameter must be a variable name, not a string value.
Delimiter	The delimiter must be a single character. When multiple characters are specified, only the first character is used. The default is '-'.
Token index	The number of the field to cut. This may also be specified as a range, with '-' sign, or a group of ranges and fields, with '&' sign. Leaving off one end of the range will result in a full range, for example: 3- would give the 3rd field and everything after that field -2 would give the 2nd field and everything before that field

FieldQty

Counts tokens separated with an arbitrary delimiter

This function counts the fields with an arbitrary delimiter and returns the resulting count.

Property Editor

Field	Description
Variable	A variable that stores fields to be count.
Delimiter	A delimiter to use when counting.

RegEx

Tests whether text matches regular expression

This function will check the text if it matches the regular expression and will return 1 if text matches regular expression.

Property Editor

Field	Description
RegEx	Regular expression.
Text	A text.

UriEncode

Encodes a string to URI-safe encoding

This function encodes a string to URI safe encoding.

Property Editor

Field	Description
URI	A URI to be used when encoding.

UriDecode

Decodes an URI-encoded string

This function decodes an URI encoded string.

Property Editor

Field	Description
URI	A URI to be used when decoding.

StrfTime

Formats the datetime

This function formats the time localized to the timezone.

Property Editor

Field	Description
Time	A time in format YYYY-MM-DD HH:MM:SS. Default is current time.
Timezone	A timezone. For example: America/Chicago. Timezone defaults to the timezone on the host computer. A list of possible timezones may be obtained from the directory listing in /usr/share/zoneinfo. The default format is %c.
Format	A format. For example: %Y-%m-%d %H:%M:%S Format may vary from one to another Asterisk distribution, as their are different implementations of strftime function. If these values do not work correctly on your system, please consult the man page for your implementation of strftime function. Here are few format strings that may be used: %a The abbreviated weekday name according to the current locale. Example: Sun %A The full weekday name according to the current locale. Example: Sunday %b The abbreviated month name according to the current locale. Example: Jul %c The preferred date and time representation for the current locale. Example: Sun Jul 22 14:57:30 2007 %d The day of the month as a decimal number (range 01 to 31). Example: 22 %F Equivalent to %Y-%m-%d (the ISO 8601 date format). Example: 2007-07-22 %H The hour as a decimal number using a 24-hour clock (range 00 to 23). Example: 14 %I The hour as a decimal number using a 12-hour clock (range 01 to 12). Example: 02 %j The day of the year as a decimal number (range 001 to 366). Example: 203 %m The month as a decimal number (range 01 to 12). Example: 07

	%M The minute as a decimal number (range 00 to 59). Example: 57 %n A newline character. %p Either 'AM' or 'PM' according to the given time value, or the corresponding strings for the current locale. Noon is treated as 'pm' and midnight as 'am'. Example: PM %r The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to '%I:%M:%S %p'. Example: 02:57:30 PM %R The time in 24-hour notation (%H:%M). For a version including the seconds, see %T below. Example: 14:57 %s The number of seconds since the Epoch, i.e., since 1970-01-01 00:00:00 UTC. Example: 1185130650 %S The second as a decimal number (range 00 to 60). (The range is up to 60 to allow for occasional leap seconds.) Example: 30 %T The time in 24-hour notation (%H:%M:%S). Example: 14:57:30 %u The day of the week as a decimal, range 1 to 7, Monday being 1. %U The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. %w The day of the week as a decimal, range 0 to 6, Sunday being 0. %x The preferred date representation for the current locale without the time. Example: 07/22/07 %X The preferred time representation for the current locale without the date. Example: 14:57:30 %y The year as a decimal number without a century (range 00 to 99). Example: 07 %Y The year as a decimal number including the century. Example: 2007 %Z The time zone or name or abbreviation. Example: EDT

Base64Encode

Encode a string in base64

This function encode a string in base64. Returns the resulting base64 string.

Base64Decode

Decode a base64 string

This function decode a base64 string. Returns the resulting plain text string.

Filter

Filter the string to include only the allowed characters

This function filter the string to include only the allowed characters. Returns the resulting string.

QuoteFunction

Quotes a given string, escaping embedded quotes as necessary

This function quotes a given string, escaping embedded quotes as necessary. Returns the quoted string.

KeypadHash

Hash the letters in the string into the equivalent keypad numbers

This function hash the letters in the string into the equivalent keypad numbers. Returns the string input, with the characters a, b, or c mapped to "2"; the characters d, e, or f mapped to "3"; etc.

StrpTime

Returns the epoch of the arbitrary date/time string structured as described in the format.

This function is useful for converting a date into an EPOCH time, possibly to pass to an application like SayUnixTime or to calculate the difference between two date strings. The function returns the string representing the unix time in seconds since the beginning of the epoch (Jan 1st, 1970)

SprintfFunction

Format a variable according to a format string

Parses the format string specified and returns a string matching that format. Supports most options supported by sprintf(3). Returns a shortened string if a format specifier is not recognized.

ToLower

Convert the string to lower case

This function converts string to lower case.

ToUpper

Convert the string to upper case

This function converts string to upper case.

Chapter

6

ASR Grammar

This document defines the syntax for grammar representation. The grammars are intended for use by speech recognizers (LumenVox Speech Engine, www.LumenVox.com) and other grammar processors so that developers can specify the words and patterns of words to be listened for by a speech recognizer.

You may refer to the W3C specification for complete details on writing grammars http://www.w3.org/TR/speech-grammar/, or to LumenVox grammar tutorial available at http://www.lumenvox.com/help/speechEngine/, but you may also find this document a useful tutorial to general principles used in writing grammars.

The syntax of the grammar format is presented in two forms, an Augmented BNF (ABNF) Form and an XML Form. Visual Dialplan supports ABNF grammar format only.

A *grammar processor* is any entity that accepts as input grammars as described in this specification. LumenVox Speech Engine (www.LumenVox.com) is grammar processor recommended to be used with Asterisk PBX.

A *user agent* is a grammar processor that accepts user input and matches that input against a grammar to produce a recognition result that represents the detected input.

The primary use of a speech recognizer grammar is to permit a speech application to indicate to a recognizer what it should listen for, specifically:

- Words that may be spoken
- Patterns in which those words may occur
- Spoken language of each word

Terminology

Requirements terms
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. However, for readability, these words do not appear in all uppercase letters in this document.

URI: Uniform Resource Identifier
A URI is a unifying syntax for the expression of names and addresses of objects on the network as used in the World Wide Web. In the ABNF Form of this specification a URI is delimited by angle brackets ("<" ">"). For example, <http://www.example.com/file-path>

Media Type
A *media type* specifies the nature of a linked resource. Media types are case insensitive. In places where a URI can be specified a media type may be provided to indicate the content type of URI. In the ABNF Form a media type may be attached as a postfix to any URI. The media type is delimited by angle brackets ("<" ">") and the URI and media type are separated by a tilde character ("~") without intervening white space. For example, <http://example.com/file-path>~<media-type>

Language identifier
A *language identifier* labels information content as being of a particular human language variant.

White space
White space consists of one or more space (#x20) characters, carriage returns, line feeds, or tabs.

DTMF
DTMF (Dual Tone Multiple Frequency) is an ITU standard for telephony signaling.

The content of this document is written according to the Speech Recognition Grammar Specification Version 1.0, a W3C-defined standard for writing grammars - http://www.w3.org/TR/speech-grammar/, copyright 2004 W3C (MIT, ERCIM, Keio), All Rights Reserved.

Rule Expansions

Under this chapter we will discuss the following rule expansions:

- Tokens
- Rule References
- Sequences and Encapsulation
- Alternatives
- Repeats
- Tags
- Language
- Precedence

A *legal rule* expansion is any legal token, rule reference, tag, or any logical combination of legal rule expansions as sequence, alternatives, repeated expansion or language-attributed expansion.

A rule expansion is formally a *regular expression*.

A *rule definition* associates a legal rule expansion with a rulename.

Tokens

A *token* (a.k.a. a terminal symbol) is the part of a grammar that defines words or other entities that may be spoken. Any legal token is a legal expansion.

Any unmarked text within a rule definition, tag content, is *token content*. The unmarked text is delimited by any syntactic construct of the grammar form. For each token content span in a grammar the grammar processor applies the following processes: *tokenization*, *white space normalization*, *token normalization* and *pronunciation lookup processes*.

Tokenization behavior: Text spans containing token sequences are delimited as follows:

- Any token in ABNF Form may be delimited by double quotes. The text contained within the double quotes is an unnormalized token. The text must not contain any double quote characters. A token delimited by double quotes may contain white space.
- Any token content not delimited by a <token> element or double quotes is treated as a sequence of white-space-delimited tokens. Each token contained in the token content is delimited at the start and at the end by any white space character or any syntactic construct that delimits a token content span.

White Space Normalization: White space must be normalized when contained in any token delimited by a <token> elements or by double quotes. Leading and trailing white space characters are stripped. Any token-internal white space character or sequence is collapsed to a single space character (#x20). For example, the following are all normalized to the same string, "San Francisco".

"San Francisco"
" San Francisco "
"San Francisco"
" San Francisco "

Because the presence of white space within a token is significant the following are distinct tokens.

"San Francisco"
"SanFrancisco"
"San_Francisco"

Token Normalization: Other normalization processes are applied to the white space normalized token according to the language and the capabilities of the speech recognizer.

Pronunciation Lookup: To match spoken (audio) input to a grammar a speech recognition must be capable of modeling the audio patterns of any token in a grammar.

Limitations of token handling: the following is informative guidance to grammar developers.

Grammar authors can improve document portability by avoiding characters and forms in tokens that do not have obvious pronunciations in the language. For English, the following are ways to handle some orthographic forms:

- Acronyms should be avoided. Alphabetic characters should be widely available. For example, replace "USA" by "u s a"; replace "W3C" by "w three c"; replace "IEEE" by "i triple e".
- Abbreviations should be replaced by the unabbreviated form. For example, replace "Dr." by "drive" or "doctor".
- Most punctuation should be expanded to a spelled form. For example replace "&" by "ampersand" or "and"; replace "+" by "plus"; replace "<" by "less than" or "open angle bracket".
- A grammar processor should support digits (e.g. "0" though "9" for European scripts). Other natural numbers should be replaced by spelled forms. For example, for US English replace "10" by "ten" and "1000" by "thousand".
- Grammar authors should consider the possibility that a grammar will be used to interpret input in a non-speech recognition device. For example, grammars can be used to process text strings from keyboard input, text telephone services, pen input and other text modalities. To facilitate text input a grammar should contain standard orthographic tokens of the language. That is, to facilitate non-speech recognition input the grammar should contain standard spellings of natural language words to the greatest extent possible.

Rule Reference

Every rule definition has a local name that must be unique within the scope of the grammar in which it is defined.

This table summarizes the various forms of rule reference that are possible within and across grammar documents.

Reference type	ABNF Form
Explicit local rule reference	$rulename
Explicit reference to a named rule of a grammar identified by a URI	$<grammarURI#rulename>
Implicit reference to the root rule of a grammar identified by a URI	$<grammarURI>
Explicit reference to a named rule of a grammar identified by a URI with a media type	$<grammarURI#rulename>~<media-type>
Implicit reference to the root rule of a grammar identified by a URI with a media type	$<grammarURI>~<media-type>
Special rule definitions	$NULL $VOID $GARBAGE

Local References

When referencing rules defined locally (defined in the same grammar as contains the reference), always use a simple rulename reference which consists of the local rulename only.
The simple rulename reference is prefixed by a "$" character.

```
$city
$digit
```

External Reference by URI

References to rules defined in other grammars are legal under some conditions. The external reference must identify the external grammar by URI and may identify a specific rule within that grammar. If the fragment identifier that would indicate a rulename is omitted, then the reference implicitly targets the root rule of the external grammar.

In ABNF an external reference by URI is represented by a dollar sign ('$') followed immediately by either an ABNF URI or ABNF URI with media type. There must be no white space between the dollar sign and the URI.

```
// References to specific rules of an external grammar
$<http://grammar.example.com/world-cities.gram#canada>
$<http://grammar.example.com/numbers.gram#digit>


// Implicit reference to the root rule of an external grammar
$<../date.gram>


// References with associated media types
$<http://grammar.example.com/world-cities.gram#canada>~<application/srgs>
$<../date.gram>~<application/srgs>
```

Special Rules

Several rulenames are defined to have specific interpretation and processing by a speech recognizer. A grammar must not redefine these rulenames.

In the ABNF Form a special rule reference is syntactically identical to a local rule reference. However, the names of the special rules are *reserved* to prevent a rule definition with the same name.

NULL
Defines a rule that is automatically matched: that is, matched without the user speaking any word.
ABNF Form: $NULLv

VOID
Defines a rule that can never be spoken. Inserting VOID into a sequence automatically makes that sequence unspeakable.
ABNF Form: $VOID

GARBAGE
Defines a rule that may match any speech up until the next rule match, the next token or until the end of spoken input. A grammar processor must accept grammars that contain special references to GARBAGE. The behavior GARBAGE rule is implementation-specific. A user agent should be capable of matching arbitrary spoken input up to the next token but may treat GARBAGE as equivalent to NULL (match no spoken input).
ABNF Form: $GARBAGE

Informative example: given suitable definitions of US cities and states, a speech recognizer may implement the following ABNF rule definitions to match "*Philadelphia in the great state of Pennsylvania*" as well as simply "*Philadelphia Pennsylvania*".

```
$location = $city $GARBAGE $state;
```

Sequences and Encapsulation

A *sequence* of legal rule expansions is itself a legal rule expansion.

ABNF Form provides syntax for encapsulating any expansion. This is used, for example, to attach a repeat operator, a language identifier or to ensure correct precedence in parsing.

A sequence of legal expansions separated by white space is a legal expansion.

A legal expansion surrounded by parentheses ("(" and ")") is a legal expansion.

```
this is a test // sequence of tokens
$action $object // sequence of rule references
the $object is $color // sequence of tokens and rule references
(fly to $city) // parentheses for encapsulation
```

Special cases
An empty parenthetical is legal as is a parenthetical containing only white space; e.g. "()" or "( )". Both forms are equivalent to $NULL and a grammar processor will behave as if the parenthetical were not present.

```
// equivalent sequences
phone home
phone ( ) home
```

Alternatives and Weights

Any set of *alternative legal rule expansions* is itself a legal rule expansion. For input to match a set of alternative rule expansions it must match one of the set of alternative expansions. A set of alternatives must contain one or more alternatives.

Any set of alternatives may be labeled with a language attachment. In the ABNF Form to ensure correct precedence the set of alternatives must be delimited by parentheses with the ABNF language attachment immediately following.

Weights

A weight may be optionally provided for any number of alternatives in an alternative expansion. Weights are simple positive floating point values without exponentials. Legal formats are "n", "n.", ".n" and "n.n" where "n" is a sequence of one or many digits.

A weight is nominally a multiplying factor in the likelihood domain of a speech recognition search. A weight of 1.0 is equivalent to providing no weight at all. A weight greater than "1.0" positively biases the alternative and a weight less than "1.0" negatively biases the alternative.

A set of alternative choices is identified as a list of legal expansions separated by the vertical bar symbol. If necessary, the set of alternative choices may be delimited by parentheses.

```
Michael | Yuriko | Mary | Duke | $otherNames
(1 | 2 | 3)
```

A weight is surrounded by forward slashes and placed before each item in the alternatives list.

```
/10/ small | /2/ medium | large
/3.1415/ pie | /1.414/ root beer | /.25/ cola
```

Special Cases

It is legal for an alternative to be a reference to $NULL, an empty parenthetical or a single tag. In each case the input is equivalent to matching $NULL and as a result the other alternatives are optional.

```
// Legal
$rule1 = word | $NULL;
$rule2 = () | word;
$rule3 = word | {TAG-CONTENT};
```

An empty alternative (white space only) is not legal.

```
// ILLEGAL
$rule1 = a | | b;
$rule2 = | b;
$rule3 = a |;
```

The following construct is interpreted as a single weighted alternative.

```
// Legal
$rule1 = /2/ word;
$rule2 = /2/ {TAG-CONTENT};
$rule3 = /2/ $NULL;
```

Repeats

Any *repeated legal rule expansion* is itself a legal rule expansion.

Operators are provided that define a legal rule expansion as being another sub-expansion that is optional, that is repeated zero or more times, that is repeated one or more times, or that is repeated some range of times.

ABNF Form Example	Behavior
<n> <6>	The contained expansion is repeated exactly "n" times. "n" must be "0" or a positive integer.
<m-n> <4-6>	The contained expansion is repeated between "m" and "n" times (inclusive). "m" and "n" must both be "0" or a positive integer and "m" must be less than or equal to "n".
<m-> <3->	The contained expansion is repeated "m" times or more (inclusive). "m" must be "0" or a positive integer. For example, "3-" declares that the contained expansion can occur three, four, five or more times.
<0-1> [...]	The contained expansion is optional.

Common Repeats

As indicated in the table above, an expansion that can occur 0-1 times is optional. Because optionality is such a common form the ABNF syntax provides square brackets as a special operator for representing optionality.

A repeat of "0-" indicates that an expansion can occur zero times, once or any number of multiple times. In regular expression languages this is often represented by the Kleene star ('*') which is reserved but not used in ABNF.

A repeat of "1-" indicates that an expansion can occur once or any number of multiple times. In regular expression languages this is often represented by the *positive closure* ('+') which is reserved but not used in ABNF.

Although ABNF supports a grammar that permits an unbounded number of input tokens it is not the case that users will speak indefinitely. Speech recognition can perform more effectively if the author indicates a more limited range of repeat occurrences.

Special Cases
Where a number of possible repetitions (e.g. <m-> or <m-n> (n > 0) but not <0>) is expressed on a construct whose only content is one or more tag elements, the behavior of the grammar processor is not defined and will be specific to individual implementations.

Any number of non-optional repetitions (e.g., <m-n>; m>0) of VOID is equivalent to a single VOID.

The behavior of a grammar processor in handling any number of repetitions of NULL is not defined and will be specific to individual implementations.

If the number of repetitions for any expansion can be only zero (i.e. <0> or <0-0>) then the expansion is equivalent to NULL.

Repeat Probabilities
Any repeat operator may specify an optional repeat probability. The value indicates the probability of successive repetition of the repeated expansion.

A repeat probability value must be in the floating pointing range of "0.0" to "1.0" (inclusive). Values outside this range are illegal. The floating point format is one of "n", "n.", "n.nnnn", ".nnnn" (with any number of digits after the dot).

Note: repeat probabilities and weights are different logical entities and have a different impact upon a speech recognition search.

Informative example: A simple example is an optional expansion (zero or one occurrences) with a probability - say "0.6". The grammar indicates that the chance that the expansion will be matched is 60% and that the chance that the expansion will not be present is 40%.

When no maximum is specified in a range (m-) the probabilities decay exponentially.

The following are postfix operators: <m-n> <m-> <m>. A postfix operator is logically attached to the preceding expansion. Postfix operators have high precedence and so are tightly bound to the immediately preceding expansion.

Optional expansions may be delimited by square brackets: [expansion]. Alternatively, an optional expansion is indicated by the postfix operator "<0-1>".

The following symbols are reserved for future use in ABNF: '*', '+', '?'. These symbols must not be used at any place in a

grammar where the syntax currently permits a repeat operator.

```
// the token "very" is optional
[very]
very <0-1>

// the rule reference $digit can occur zero, one or many times
$digit <0->

// the rule reference $digit can occur one or more times
$digit <1->

// the rule reference $digit can occur four, five or six times
$digit <4-6>

// the rule reference $digit can occur ten or more times
$digit <10->

// Examples of the following expansion
// "pizza"
// "big pizza with pepperoni"
// "very big pizza with cheese and pepperoni"
[[very] big] pizza ([with | and] $topping) <0->
```

Repeat probabilities are only supported in the range form. The probability is delimited by slash characters and contained within the angle brackets: <m-n /prob/> and <m- /prob/>.

```
// the token "very" is optional and is 60% likely to occur
// and 40% likely to be absent in input
very <0-1 /0.6/>
// the rule reference $digit must occur two to four times
// with 80% probability of recurrence
$digit <2-4 /.8/>
```

Tags

A *tag* is a legal rule expansion (a tag can also be declared in the grammar header).

A tag is an *arbitrary string* that may be included inline within any legal rule expansion. Any number of tags may be included inline within a rule expansion.

Tags do not affect the legal word patterns defined by the grammars or the process of recognizing speech or other input given a grammar.

Tags may contain content for semantic interpretation. The semantic interpretation processes may affect the recognition result.

Language attachments have no effect upon tags.

The tag format declaration indicates the content type of all tags in a grammar.

Special Cases
It is legal to use a tag as a stand-alone expansion. For example, a rule may expand to a single tag and no tokens.

```
$rule = {TAG-CONTENT};
```

A tag is delimited by either a pair of opening and closing curly brackets - '{' and '}' - or by the following 3-character sequences which are considered very unlikely to occur within a tag - '{!{' and '}!}'. A tag delimited by single curly brackets cannot contain the single closing curly bracket character ('}'). A tag delimited by the 3-character sequence cannot contain the closing 3-character sequence ('}!}').

The tag content is all text between the opening and closing character sequences including leading and trailing white space. The contents of the tag are not parsed by the grammar processor.

Tag precedence is the same as for rule references and tokens. In the first example below there is a sequence of six space-separated expansions (3 tokens, a tag, a token and a tag). In the second example, the alternative is a choice between a sequence containing a token and a tag or a sequence containing a rule reference and a tag.

```
$rule1 = this is a {TAG-CONTENT-1} test {TAG-CONTENT-2};

$rule2 = open {TAG-CONTENT-1} | $close {TAG-CONTENT-2};

$rule3 = {!{ a simple tag containing { and } needs no escaping }!};
```

Language

Any legal rule expansion that has an attached language identifier is itself a legal rule expansion. ABNF permit a legal language identifier to be attached to any token, sequence or set of alternatives

The language declaration for a rule expansion affects only the contained content. Moreover, the language declaration affects only the handling of tokens in the contained content and does not affect tags or rule references.

By default a grammar is a single language document with a language identifier provided in the language declaration in the grammar header. All tokens within that grammar, unless otherwise declared, will be handled according to the grammar's language.

In the ABNF Form a language identifier may be right-attached to any legal rule expansion except rule reference. The attachment is an exclamation point character ('!') followed by a legal language identifier without intervening white space.

The language attachment has higher precedence than sequences or alternatives. To attach a language to these rule expansion types the expansion should be delimited by parentheses.

```
#ABNF 1.0 ISO-8859-1;

// Default grammar language is US English
language en-US;

// Single language attachment to tokens
// Note that "fr-CA" (Canadian French) is applied to only
// the word "oui" because of precedence rules
$yes = yes | oui!fr-CA;

// Single language attachment to an expansion
$people1 = (Michel Tremblay | Andre Roy)!fr-CA;

// Handling language-specific pronunciations of the same word
// A capable speech recognizer will listen for Mexican Spanish and
// US English pronunciations.
$people2 = Jose!en-US; | Jose!es-MX;

/** * Multi-lingual input possible
* @example may I speak to Andre Roy
* @example may I speak to Jose
*/
public $request = may I speak to ($people1 | $people2);
```

Precedence

This section defines the precedence of the ABNF rule expansion syntax.

The precedence definitions for the ABNF Form are intended to minimize the need for parentheses.

The following is the ordering of precedence of rule expansions. Parentheses may be used to explicitly control rule structure.

1. A rule reference, a quoted token, an unquoted token or a tag.

2. Parentheses ('(' and ')') for grouping and square brackets ('[' and ']') for optional grouping.

3. Repeat operator (e.g. "<0-1>") and language attachment (e.g. "!en-AU") apply to the tightest immediate preceding rule expansion. (To apply them to a sequence or to alternatives, use `()' or `[]' for grouping.)

4. Sequence of rule expansions.

5. Set of alternative rule expansions separated by vertical bars ('|') with optional weights.

Phonetic Spellings

You can specify pronunciations of words using phonemes. Phonemes are the basic sounds that make up words. Specifying pronunciations this way is useful for adding alternative pronunciations to help deal with some proper names and to help support dialects or even foreign words.

To add a phonetic spelling to a rule, enclose the phonetic spelling within double quotation marks " " and curly braces { }.

For instance, the word "either" is commonly pronounced in two ways. One has a long I sound at the start (eye-ther) while the other starts with a long E sound (ee-ther). Broken into their phonemes, these two variants would be spelled as AY DH AXR and IY DH AXR.

A rule that contained the two variants of the word "either" might look like this:

```
$either = "{AY DH AXR}" | "{IY DH AXR}";
```

A rule set up like this, however, will only return the phonemes if the rule is matched. If you want to get the actual word returned as raw text by the Engine, you need to enclose the word within the quotation marks and curly braces, but separate it from the phonemes with a colon.

```
$either = "{AY DH AXR:either}" | "{IY DH AXR:either}";
```

Note that adding phonetic spellings is distinct from semantic interpretation, even though both use curly braces in ABNF (the curly braces for a phonetic spelling are always inside of double quotation marks).

Here is a Spanish grammar that uses these principles:

```
#ABNF 1.0 ISO-8859-1;
/*
* This is an example name grammar
* with semantic interpretation tags.
*/

language es-MX;
mode voice;
root $main;
tag-format <semantics/1.0.2006>
public $main = "{K R EY D IY T OW:credito}"
{out="credito"} //custom pronunciation
| Si {out="Si"}
| (("el ba?o")
| "{EY L BV AA GN OW}") {out="el ba?o"}
| hola {out="hola"};
```

Foreign Words

Using phonetic rules within an grammar, developers can build limited functionality for words in languages not supported by the speech recognizers.

Within a grammar you may create rules that are matched to phonetic spellings by using phonemes. By breaking foreign words into their constituent phonemes, grammars can effectively contain those words.

If a word contains a phoneme that does not occur in English, use the English phoneme that is closest to the foreign phoneme.

For instance, if you were going to build phonetic support for the German numbers zero through ten, you would begin by assembling a list of the digits along with phonetic pronunciations:

Number	Phonemes
Null	N UH L
Eins	AY N S
Zwei	S V AY | T S V AY
Zwo	S V AY | T S V AY
Drei	D R AY
Vier	F IY R
F?nf	F UW N F
Sechs	Z EH K S
Sieben	Z IY B EH N
Acht	AA K T
Neun	N OY N

Note that several of the phonemes are approximations. The "ch" sound in "acht" is different from the English phoneme represented by K, but since English does not have the phoneme from "acht" we pick the closest sound.

To enter a phonetic spelling as part of a rule within an grammar, include the phonetic spelling inside of quotes and curly braces. A basic ABNF grammar to return a spoken German digit might look like the following:

```
#ABNF 1.0;
language en-US;
mode voice;
tag-format <lumenvox/1.0>

root $Digits;

$Digit = ("{AY N S: eins}":"1" |
("{S V AY: zwei}" | "{T S V AY: zwei}"):"2" |
"{D R AY: drei}":"3" |
"{F IY R: vier}":"4" |
"{F UW N F: funf}":"5" |
"{Z EH K S: sechs}":"6" |
"{Z IY B EH N: sieben}":"7" |
"{AA K T: acht}":"8" |
"{N OY N: neun}":"9" |
"{N UW L: null}":"0");
$Digits = {$=''} ($Digit {$+=$$})<1->;
```

Rule Definitions

A *rule definition* associates a legal rule expansion with a *rulename*. The rule definition is also responsible for defining the *scope* of the rule definition: whether it is local to the grammar in which it is defined or whether it may be referenced within other grammars. Finally, the rule definition may additionally include documentation comments and other pragmatics.

The rulename for each rule definition must be unique within a grammar. The same rulename may be used in multiple grammars.

A rule definition is referenced by a URI in a rule reference with the rulename being represented as the fragment identifier.

Basic Rule Definition
The core purpose of a rule definition is to associate a legal rule expansion with a rulename.

Defined rulenames must be unique within a grammar. Rulenames are case-sensitive. Exact string comparison is used to resolve rulename references.

A legal rulename cannot be one of the special rules: specifically "NULL", "VOID" or "GARBAGE".

The rule definition consists of an optional scoping declaration (explained in the next section) followed by a legal rule name, an equals sign, a legal rule expansion and a closing semicolon. The rule definition has one of the following legal forms:

```
$ruleName = ruleExpansion;
public $ruleName = ruleExpansion;
private $ruleName = ruleExpansion;
```

For example:

```
$city = Boston | "New York" | Madrid;
$command = $action $object;
```

Special Cases
An empty rule definition is illegal.

It is legal to define a rule that expands to empty parentheses or $NULL (equivalent forms). It is legal to define a rule that expands to a single tag.

```
// Legal
$rule = ();
$rule = $NULL;
$rule = {TAG-CONTENT};

// ILLEGAL
$rule = ;
```

Scoping of Rule Definitions
Each defined rule has a scope. The scope is either "private" or "public". If not explicitly declared in a rule definition then the scope defaults to "private".

A public-scoped rule may be explicitly referenced (using the fragment identifier syntax of a URI) in the rule definitions of other grammars and in other non-grammar documents. A private-scoped rule cannot be so referenced and is directly accessible only within its containing grammar. A private rule may be explicitly referenced only by other rules within the same grammar.

A rule definition may be annotated with the keywords "public" or "private". If no scope is provided, the default is "private".

```
$town = Townsville | Beantown;
private $city = Boston | "New York" | Madrid;
public $command = $action $object;
```

Example Phrases
It is often desirable to include examples of phrases that match rule definitions along with the definition. Zero, one or many example phrases may be provided for any rule definition. Because the examples are explicitly marked, automated tools can be used for regression testing and for generation of grammar documentation.

A documentation comment is a C/C++/Java comment that starts with the sequence of characters /** and which immediately precedes the relevant rule definition. Zero or more @example tags may be contained at the end of the documentation comment.

```
/**
* A simple directive to execute an action.
*
* @example open the window
* @example close the door
*/
public $command = $action $object;
```

Semantic Interpretation

When building an application using speech recognition, it is often not enough to know what the user said but rather we need the meaning of an utterance. One caller may ask for "Customer Support" while another asks for "Customer Service," but as far as a call router is concerned, these are really the same thing.

Assigning meaning from a raw text is a process called semantic interpretation.

Creating a grammar and examining the parse tree generated by the Engine from a user's speech is the first step toward semantic interpretation. But it is often not enough to just read off the values of the tree; significant post processing of the tree is necessary to extract meaning.

Here is an ABNF grammar that matches speaking numbers from zero to nine hundred and ninety nine (it is by no means complete; for instance, it cannot recognize "two forty six" for 246):

```
#ABNF 1.0;
language en-US;
mode voice;
root $small_number;


$base = one | two | three | four | five | six | seven | eight | nine;
$teen = ten | eleven | twelve | thirteen | fourteen | fifteen |
sixteen | seventeen | eighteen  | nineteen;
$twenty_to_ninetynine = (twenty | thirty | forty | fifty | sixty |
seventy | eighty  | ninety)[$base];
$tens = $base | $teen | $twenty_to_ninetynine;
$hundred = ([a] hundred  |  $base hundred);
$small_number = $hundred [[and] $tens]  |  $tens;
```

If the Engine recognizes "two hundred twelve," the parse tree looks like this:

```
$small_number:
   $hundred:
      $base:
         "TWO"
      "HUNDRED"
   $tens:
      $teen:
         "TWELVE"
```

But if your application needs to determine if a speaker spoke a number larger than 500, then it's not enough to know the parse tree; all you have is a structure of words. You need to write code to transform the tree into the number 212.

The logic to do this transformation is going to be tied closely to the grammar's rules. For instance, within the $hundred rule, you have to know that there is an optional $base rule that has to be multiplied by 100. But in the $twenty_to_ninetynine rule, the optional $base has to be added to the total of the number you are building.

Because of the close relationship between a grammar's rules, and the semantic interpretation process, it can be convenient if you can put the semantic interpretation directly into the grammar.

Using a standard called Semantic Interpretation for Speech Recognition, grammars can contain logic that can aid in semantic interpretation. Using SISR, you place "tags" into grammars.

Semantic Interpretation for Speech Recognition

Tags

Tags are how Semantic Interpretation for Speech Recognition (SISR) is put into grammars. Tags usually contain ECMAScript. Any script inside of a tag is executed by the Engine when the part of a rule to its left is matched.

In an ABNF grammar, tags are noted by curly brackets: **{** and **}**. Because ECMAScript also uses the curly bracket as a reserved character for conditional statements, it is sometimes necessary to have curly braces within an SISR tag. In this case, you can denote an SISR tag with the following three-character sequence: **{!{** will open a tag and **}!}** will close it.

The following two grammar examples are equivalent. Both listen for the word "hello" and return the string "Hello, world." when matched.

```
$hello = hello {out = "Hello, world.";};
```

Or, using the three-character identifier:

```
$hello = hello {!{out = "Hello, world.";}!};
```

If there are multiple tags in a matched rule, they are executed in left-to-right order. If a rule contains multiple alternatives, tags enclosed within an alternative are executed only if the alternative is matched. For example:

```
$yesorno = yes {out = "Yes"} | no {out = "No"};
```

If the user says "yes" only the first tag is executed, as the alternative was not matched. But consider another example:

```
$yesorno = {out = "Definitely"} (yes {out += "Yes"} | no {out += "No"});
```

In this case, if the user says "yes," both the first and second tags are executed, as the {out = "Definitely"} tag was not restricted to either alternative. Likewise if the user says "no," the first and third tags are executed.

Rule Variables

Each rule in the grammar has a variable associated with it. This variable is called the ***rule variable***. The SISR tags in a rule modify that rule's variable, and the rule variable for the grammar's root rule is what is returned as the interpretation for the grammar.

If a rule has no SISR specified, its variable is set to the raw text of the utterance, i.e. whatever the speaker said that matched the rule. As an example, consider the following rule:

```
$robertsmith = (robert | bob) [smith];
```

The rule is matched if the speaker says "Robert," "Robert Smith," "Bob," or "Bob Smith." Since no SISR is associated with the rule, if the caller says "Robert Smith" the variable is set to "Robert Smith." If the caller says "Bob," the rule variable is set to "Bob."

Ultimately, the Engine returns the rule variable from a grammar's root rule. So in a sense, writing SISR is about writing tags that will affect the root rule's variable.

Tag Formats

There are two types of SISR. The first, called ***String Literals*** does not use ECMAScript. Instead, anything within a tag is put directly into the rule's variable as a string. For instance, using String Literals with the following rule would look like this:

```
$robertsmith = (robert | bob) [smith] {Robert Smith};
```

Now, regardless of whether a speaker says "Robert," "Robert Smith," "Bob," or "Bob Smith," the grammar returns "Robert Smith" as a string.

This simplicity makes string literals a good choice for very simple applications where no logic needs to be performed on semantic interpretation information. For many applications, however, String Literals is not sufficient.

The downside of string literals is that there is no way to perform any sort of logical operations on rule variables. One cannot add variables, and thus if two rules in a grammar are matched there is no way to preserve the SISR information.

For this reason, ***SI Script*** is the other type of SISR. In this type, each tag contains ECMAScript that is executed when the tag's rule is matched. Though a little more cumbersome to write, SI Script allows for great control over rule variables.

You cannot mix tag formats within a single grammar.

A grammar that uses SISR must specify which tag format it is using in the header. In ABNF, this is done with a line called ***tag-format***.

ABNF Header Example:

```
#ABNF 1.0 UTF-8;
language en-US;
mode voice;
tag-format <semantics/1.0.2006>;
```

Rule Variables

Once you understand SISR Basics, it is time to start actually putting SISR tags into action.

SISR is concerned mainly with accessing and modifying rule variables. The ultimate return of a grammar is the rule variable of the root rule. How rule variables are modified depends on whether one is using string literals or SI Script.

String Literals
In grammars using the string literals format, anything in a tag is put into the rule variable for a current rule. There is no way to perform any sort of operations. In order to set the root rule's variable, you must either explicitly put the entire desired return result in the root rule or else rely on inheritance.

Rule Visibility and Inheritance
When a rule references another rule that is matched, the referenced rule becomes a child of the original rule. Consider a very simple grammar using the string literals tag format:

```
#ABNF 1.0 UTF-8;
language en-US;
mode voice;
tag-format <semantics/1.0.2006-literals>;

root $boolean;

$boolean = $yes | $no;
$yes = yes {true};
$no = no {false};
```

If a speaker says "yes" the parse tree looks like:

$boolean $yes

The $yes rule is a child rule of $boolean. Only the direct children of a rule are "visible" to the parent rule, meaning that a rule can only access the variables of its children rules.

If no semantic interpretation tags are present in a matched rule, that rule's variable will inherit the value of the variable of the last visible rule. So in the example above, $boolean has no SISR tags, so it inherits the value of $yes, which in this

case is the string "true."

Note that is good practice to not rely too heavily on inheritance, as it can be difficult to troubleshoot. It is often easier to explicitly give each rule SISR tags.

SI Script

Inheritance and visibility is even more important when using SI Script, which does allow for any sort of operations that can be performed in ECMAScript (most commonly known by its JavaScript implementation).

The current rule's variable is identified by the object called **out**. It is modified using the standard ECMAScript operators. All variables are initialized as empty objects, but can be converted to strings or integers on the fly by setting their values to those types. For instance, this duplicates the grammar from above using SI Script:

```
#ABNF 1.0 UTF-8;
language en-US;
mode voice;
tag-format <semantics/1.0.2006>;

root $boolean;  $boolean = $yes |
$no; $yes = yes {out="true"};
$no = no {out="false"};
```

This grammar sets the value of out to be a string (either "true" or "false") and the root rule's variable still derives its value via inheritance.

Variable Types

Rule variables can be of any type available in ECMAScript. Until they have values assigned to them, they are not explicitly typed. Most variables will be either strings or integers, but you also may use more structured objects. You can create these objects by simply assigning values to properties. You should refer to the ECMAScript specification for more information on working with variables and variable types.

It is important to remember how operations affect data types. For instance, adding two strings combines them into a single string, while adding two integers is treated as mathematical addition on two numbers.

Accessing Other Rule Variables

One of the powers of SI Script is the ability to access the value of any visible rule. This is necessary for concatenating strings or performing mathematical operations. For instance, a digits grammar generally expects that rules will be matched multiple times. One way of building this sort of grammar is to loop through a rule multiple times and add to a variable with each iteration. Consider the following grammar, designed to capture an indefinitely long string of ones and zeroes:

```
root $binary;
$binary = ($one | $zero) <1->;
$one = one;
$zero = zero;
```

The difficulty here is that even if we add SISR tags to the $one and $zero rule, we need to concatenate them in the root rule. SI Script provides us with an object called ***rules*** that contains, as properties, the variables of all currently visible rules. Using ***rules.rulename***, where rulename is the name of the rule variable we want to access, we can read the values of other rule variables. One way to add SISR to this grammar is as follows:

```
root $binary;
$binary = {out=""} ($one {out+=rules.one} | $zero
{out+=rules.zero}) <1->;
$one = one {out="1"};
$zero = zero {out="0"};
```

In the $binary rule, we are doing two things that require some explanation. First, we initialize ***out*** as an empty string. Second, whenever $one or $zero is matched, we add the value of ***rules.one*** or ***rules.zero*** to the current rule variable (out). Thus if a caller says "one one zero" the return is the string "110."

The ***rules*** object provides a useful shortcut in the ***latest()*** property. ***rules.latest()*** always refers to the latest rule matched. This can save time when a rule contains many alternatives or when it is impossible to know the exact name of the last rule matched. Here is a modified version of the binary grammar:

```
root $binary;
$binary = (($one | $zero) {out+=rules.latest()})<1->;
$one = one {out="1"};
```

```
$zero = zero {out="0"};
```

In this implementation, there is no way to tell from the $binary rule whether the $one or the $zero alternatives were matched, but using rules.latest() provides the same functionality as the example before it.

Meta Variables

SI Script provides one other feature that is useful for advanced development. Each rule has associated with it a meta variable that provides information from the Speech Engine about how the rule was matched. The meta object functions similarly to the ***rules*** object: specify ***meta.rulename*** to access the meta variable for a specific rule, or ***meta.latest()*** for the last rule matched. There is also a ***meta.current()*** for the current rule (note that there is no ***rules.current()***; the equivalent there is ***out***).

There are two meta properties that can be accessed: ***text*** and ***score***. ***Text*** returns the raw text from the Engine that caused the Engine to match a rule (i.e. what the Engine recognized the user saying). ***Score*** provides the confidence score for that text (i.e. how likely it is that what the user said corresponds with the raw text).

As an example, consider the following rule:

```
$yes = yes {out.interpretation="true";
out.confidence=meta.current().score;
out.rawtext=meta.current().text;};
```

If matched, it returns an object with three properties: interpretation, confidence, and rawtext. The interpretation property contains "***true***", the confidence property contains a confidence score supplied by the Engine, and the rawtext property contains "***yes***".

SI Script by Example

You will probably find it helpful to have already read Rule Variables and the previous pages in the Semantic Interpretation before following along with this example.

In this page, using the numbers grammar from the introduction, we will look at getting semantic interpretation from a grammar.

Tag Format Declaration

```
#ABNF 1.0;
language en-US;
mode voice;
tag-format <emantics/1.0.2006>;
```

In our grammar's header, we specify ***semantics/1.0.2006*** as our tag format. This tells the Engine that we will be using the 2006 version of SISR 1.0, using SI Script tags (as opposed to string literals). Note that this deviates from the current W3C standard, which would use just ***semantics/1.0***. In order to maintain backwards compatibility with older drafts, LumenVox requires the use of 1.0.2006 to access the final version of 1.0.

Setting Semantic Interpretation
As we build our grammar rules, we need to set the rule variable for any rule that is matched:

```
$base = (one {out = 1}  | two {out = 2}| three {out = 3}|
four{out = 4}| five {out = 5}| six {out = 6}| seven {out = 7}|
eight {out = 8}  | nine {out = 9});


$teen = ten {out = 10}| eleven {out = 11}| twelve {out =
12}| thirteen {out = 13}| fourteen {out = 14}| fifteen{out =
15}| sixteen {out = 16}| seventeen {out = 17}| eighteen {out
= 18}| nineteen {out = 19};
```

The identifier ***out*** serves as the variable name for the current rule's variable. We are setting the values of the variables for $base and $teen to the numerical representation of the word that was spoken. We are specifying the values as integers.

Modifying Rule Variables

We can perform operations on the variable identified by out the same way we can any other variable. We also access the rule variables for other rules by using ***rules.rulename***, where ***rulename*** is the name of the rule whose variable we want to access.

In the $twenty_to_ninetynine rule, we need to add the value of the variable from the $base rule to the rule variable for $twenty_to_ninetynine:

```
twenty_to_ninetynine = (twenty {out = 20}|thirty {out =
30}|forty {out = 40}|fifty {out = 50}|sixty {out = 60} |
seventy {out = 70}| eighty {out = 80}|ninety {out = 90})
[$base { out += rules.base }];
```

First, we sent ***out*** equal to 20, 30, 40, etc., up to ninety. Then, if the optional $base rule is matched, the value of $base is added to ***out***.

The rules.latest() Object

So far we have seen that a rule's variable can be referenced by ***rules.rulename*** after that rule has been matched. Sometimes, when there are lots of rule alternatives in a rule, it can be cumbersome to reference rules by name. Other times, a matched rule can't be referenced at all. For these reasons, the ***rules.latest()*** object exists. The ***rules.latest()*** object is always equal to the last rule matched. Using ***rules.latest()***, we can write the $tens, $hundred, and $small_number rules like this:

```
$tens = ($base|$teen|$twenty_to_ninetynine) { out =
rules.latest() };


$hundred = ([a] hundred {out = 100} | $base hundred {out
= 100 * rules.base});


$small_number = $hundred {out = rules.latest()} [[and]
$tens {out += rules.latest}] | $tens { out = rules.latest() };
```

Composite Return Types

Our small numbers grammar now returns an integer named ***small_number***. Sometimes, however, we want more than one piece of information for a return type. By default, grammar rules return an object type, and object types can have additional properties.

For example, we may also want to know the exact phrase that was spoken, possibly for transcription or reading the text back to the speaker. Each rule reference also has a corresponding meta variable, with a property called "text."

We access meta variables similarly to how we access variables using ***rules.rulename***. To get the text for a specific rule, use ***meta.rulename.text***. To get the text for the current rule, use ***meta.current().text***.

The following change to our grammar creates a composite return type containing the text that was spoken, and the numeric representation of that text.

```
root $small_number_and_text;?


$small_number_and_text = $small_number {out.number =
rules.latest();
out.text = meta.current().text};
```

Now a successful grammar match returns an object with two member properties, number and text.

Here is the complete grammar:

```
#ABNF 1.0;
language en-US;
mode voice;
tag-format <semantics/1.0.2006>;
root $small_number_and_text;
$base = (one {out = 1}  | two {out = 2} | three {out = 3} |
four{out = 4} | five {out = 5} | six {out = 6} | seven {out = 7} |
eight {out = 8}  | nine {out = 9});
$teen = ten {out = 10} | eleven {out = 11} | twelve {out =
12} | thirteen {out = 13} | fourteen {out = 14} | fifteen{out =
15} | sixteen {out = 16} | seventeen {out = 17} | eighteen {out
= 18} | nineteen {out = 19};


$twenty_to_ninetynine = (twenty {out = 20} | thirty {out =
30} | forty {out = 40} | fifty {out = 50} | sixty {out = 60} | seventy
{out = 70} | eighty {out = 80} | ninety {out = 90}) [$base { out
+= rules.base }];


$tens = ($base | $teen | $twenty_to_ninetynine) { out =
rules.latest() };


$hundred = ([a] hundred {out = 100}  |  $base hundred {out
= 100 * rules.base});
```

```
$small_number = $hundred {out = rules.latest()} [[and]
$tens {out += rules.latest}] | $tens { out = rules.latest() };


$small_number_and_text = $small_number {out.number =
rules.latest(); out.text = meta.current().text};
```

Grammar Document

A conforming stand-alone grammar document consists of a legal header followed by a body consisting of a set of legal rule definitions. All rules defined within that grammar are scoped within the grammar's rulename namespace and each rulename must be legal and unique.

It is legal for a grammar to define no rules. The grammar cannot be used for processing input since it defines no patterns for matching user input.

- Grammar Header Declarations
- Language
- Mode
- Root Rule
- Tag Format
- Base URI
- Tag
- Comments
- ABNF Keywords
- Built In Grammars

Grammar Header Declarations

A legal header for a stand-alone ABNF document consists of a required ABNF self-identifying header including the grammar version and optional character encoding followed by these declarations in any order:

- Language
- Mode
- Root rule
- Tag format
- Base URI
- Pronunciation lexicon (any number)
- Meta and http-equiv (any number)
- Tag (any number)

ABNF comments may appear between the declarations in the ABNF header after the ABNF self-identifying header. The header declarations are followed by the rule definitions of the grammar.

The following are two examples of ABNF headers. Note that ordering of the declarations (except the ABNF self-identifying header) is unimportant.

```
#ABNF 1.0 ISO-8859-1;

language en;
mode voice;
root $myRule;
tag-format FORMAT-STRING;
base <http://www.example.com/base-file-path>;
lexicon <http://www.example.com/lexicon.file>;
lexicon <http://www.example.com/strange-city-
names.file>~<media-type>;
meta "Author" is "Stephanie Williams";
http-equiv "Date" is "Fri, 10 Feb 2002 17:27:21 GMT";
{var x=1};
```

```
#ABNF 1.0;

// A French Canadian grammar
language fr-CA;

// It's a speech grammar
mode voice;

// Here's the root rule
root $QuebecCities;
```

Language

The language declaration of a grammar provides the language identifier that indicates the primary language contained by the document and optionally indicates a country or other variation. Additionally, any legal rule expansion may be labeled with a language identifier.

The language declaration is required for all speech recognition grammars: i.e. all grammars for which the mode is "voice".

In DTMF grammars a language declaration must be ignored if present.

The ABNF header must contain zero or one language declaration. It consists of the keyword "language", white space, a legal language identifier, optional white space and a terminating semicolon character (';').

```
language en-US;
```

The LumenVox Speech Engine supports the following languages:

en-US	American English acoustic model and dictionary
en-US-di	American English digits-only model
en-AU	Australian English acoustic model and dictionary
en-AU-di	Australian English digits-only model
en-GB	U.K. English acoustic model and dictionary
en-GB-di	U.K. English acoustic model and dictionary
es-MX	Mexican Spanish acoustic model and dictionary
es-MX-di	Mexican Spanish digits-only model
es-CO	South American Spanish acoustic model and dictionary
es-CO-di	South American Spanish digits-only model
fr-CA	French Canadian acoustic model and dictionary
fr-CA-di	French Canadian digits-only model

Grammar Mode

The mode of a grammar indicates the type of input that the user agent should be detecting. The default mode is "voice" for speech recognition grammars. An alternative input mode is "dtmf" input.

The mode attribute indicates how to interpret the tokens contained by the grammar. Speech tokens are expected to be detected as speech audio that sounds like the token.

The ABNF header must contain zero or one mode declaration. It consists of the keyword "mode", white space, either "voice" or "dtmf" optional white space and a terminating semicolon character (';'). If the ABNF header does not declare the mode then it defaults to voice.

```
mode voice;
mode dtmf;
```

Root Rule

The ABNF header must contain zero or one root rule declaration. It consists of the keyword "root", white space, the legal rulename of a rule defined within the grammar prefixed by the dollar sign ('$'), optional white space and a terminating semicolon character (';'). If the ABNF header does not declare the root rule then it is not legal to implicitly reference the grammar by its root.

```
root $rulename;
```

Tag Format

You may place pieces of data called tags anywhere in a grammar rule. When a rule is matched, the tag is returned to the user in a parse tree, along with the words spoken that caused the rule to match.

A common use for tags is to transform a speakers sentence into data that your application can understand.

The LumenVox speech port is capable of manipulating the tags in your parse tree, if they are in a form known as the Semantic Interpretation for Speech Recognition (SISR) tag format.

To do any kind of interpretation, you must specify the format your tags are in.

Within the speech port, the following tag format specifiers are acceptable. Currently, both formats tell the engine to perform the same interpretation process, but as other interpretation schemes are adopted, or interpretation schemes are modified, the tag format specifier you decide on will become more important.

semantics/1.0.2006	Use the SI script format of the SISR official recommendation, version 1.0 (adopted as a recommendation April 2007).
semantics/1.0.2006-literals	Use the string literals format of the SISR official recommendation, version 1.0 (adopted as a recommendation April 2007).
semantics/1.0	Use the 2003 working draft of SISR.
lumenvox/1.0	Use the LumenVox-specific implementation of the 2003 SISR draft.

If the tag format of your grammar does not match one of these specifiers, the speech port will not attempt to interpret your tags. You can still use the tag data in the Parse Tree to perform your own interpretation.

To specify the format of the tags in a grammar, use the following syntax:

```
tag-format <lumenvox/1.0>;
```

Base URI

Relative URIs are resolved according to a base URI, which may come from a variety of sources. The base URI declaration allows authors to specify a document's base URI explicitly.

The path information specified by the base URI declaration only affects URIs in the document where the element appears.

The ABNF header must contain zero or one base URI declaration. It consists of the keyword "base", white space, a legal ABNF URI, optional white space and a terminating semicolon character (';').

```
base <http://www.example.com/base-file-path>;
```

```
base <http://www.example.com/another-base-file-path>;
```

Resolving Relative URIs

User agents must calculate the base URI for resolving relative URIs according to [RFC2396]. The following describes how RFC 2396 applies to grammar documents.

User agents must calculate the base URI according to the following precedences (highest priority to lowest):

- The base URI is set by the xml:base attribute on the grammar element or the base declaration in the ABNF header
- The base URI is provided in a meta declaration
- The base URI is given by metadata discovered during a protocol interaction, such as an HTTP header (see [RFC2616]).
- By default, the base URI is that of the current document. Not all grammar documents have a base URI (e.g., a valid grammar document may appear in an email and may not be designated by a URI). Such grammar documents are not valid if they contain relative URIs and rely on a default base URI.

Tag

A grammar may optionally specify one or more tag declarations in the header. The content of a tag in the header, just like a tag in rule expansions, is an arbitrary string which may be used for semantic interpretation.

The ABNF header may contain any number of tag declarations (zero, one or many).

The tag declaration consists a string delimited as described above followed by a closing semicolon (';').

The tag content is all text between the opening and closing delimiters including leading and trailing whitespace. The contents of the tag are not parsed by the grammar processor.

```
#ABNF V1.0 ISO-8859-1;
language en-US;
{TAG-CONTENT-1};
{!{TAG-CONTENT-2}!};

$rule = . . .;
...
```

Comments

Comments may be placed in most places in a grammar document. For ABNF there are documentation comments and C/C++/Java-style comments.
C/C++/Java comments are permitted. Documentation comments are permitted before grammar and language declarations and before any rule definition.

```
// C++/Java-style single-line comment
/* C/C++/Java-style comment */
/** Java-style documentation comment */
```

ABNF Keywords

ABNF keywords are case sensitive. The keywords of the ABNF language are not reserved. The keywords with specified meaning in ABNF are:

Context	Keywords
Language declaration	"language"
Mode declaration	"mode"
Root declaration	"root"
Tag format declaration	"tag-format"
Base URI declaration	"base"
Pronunciation lexicon	"lexicon"
Meta or HTTP-equiv declaration	"meta", "http-equiv", "is"
Rule definition	"public", "private"

Since keywords are not reserved they may be used as rulenames and as tokens. The following is a legal grammar that accepts as input a sequence of one or more "public" tokens.

```
#ABNF 1.0 ISO-8859-1;

language en-AU;
root $public;
mode voice;

public $public = public $public | public;
```

Built-in Grammars

LumenVox provides the built-in grammars expected by the users.

All of them provide the required output format

URI	Sample Input	Output
builtin:grammar/boolean	"yes", "no thank you", etc.	"true" or "false"
builtin:grammar/date	"january thirteenth" or "december first two thousand"	"????0113" or "20001201"
builtin:grammar/digits	"one two three four"	"1234"
builtin:grammar/currency	"eighteen dollars and four cents"	"USD18.04"
builtin:grammar/number	"four hundred point five"	"400.5"
builtin:grammar/phone	"area code eight five eight seven oh seven oh seven oh seven"	"8587070707"
builtin:grammar/time	"six o clock" or "five thirty p m"	"0600?" or "0530p"

DTMF Grammars

This section defines a normative representation of a grammar consisting of DTMF tokens. A DTMF grammar can be used by a DTMF detector to determine sequences of legal and illegal DTMF events.

If the grammar mode is declared as "dtmf" then tokens contained by the grammar are treated as DTMF tones (rather than the default of speech tokens).

There are sixteen (16) DTMF tones. Of these twelve (12) are commonly found on telephone sets as the digits "0" through "9" plus "*" (star) and "#" (pound). The four DTMF tones not typically present on telephones are "A", "B", "C", "D".

Each of the DTMF symbols is a legal DTMF token in a DTMF grammar. As in speech grammars, tokens must be separated by white space in a DTMF grammar. A space-separated sequence of DTMF symbols represents a temporal sequence of DTMF entries.

In the ABNF Form the "*" symbol is reserved so double quotes must always be used to delimit "*" when defining an ABNF DTMF grammar. It is recommended that the "#" symbol also be quoted. As an alternative the tokens "star" and "pound" are acceptable synonyms.

In any DTMF grammar any language declaration in a grammar header is ignored and any language attachments to rule expansions are ignored.

In all other respects a DTMF grammar is syntactically the same as a speech grammar. For example, DTMF grammars may use rule references, special rules, tags and other specification features.

The following is a simple DTMF grammar that accepts a 4-digit PIN followed by a pound terminator. It also permits the sequence of "*" followed by "9" (e.g. to receive a help message).

```
#ABNF 1.0 ISO-8859-1;

mode dtmf;
$digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;
public $pin = $digit <4> "#" | "*" 9;
```

ABNF grammar examples

In this document three grammar examples will be presented:

- Simple Examples
- Cross-Reference Examples
- Phone Number grammar

Simple Examples

The following shows a simple grammar that supports commands such as "open a file" and "please move the window". It references a separately-defined grammar for politeness which is not shown here.

```
#ABNF 1.0 UTF-8;

language en;
mode voice;
root $basicCmd;

meta "author" is "Stephanie Williams";

/**
* Basic command.
* @example please move the window
* @example open a file
*/

public $basicCmd =
$<http://grammar.example.com/politeness.gram#startPolite>
$command
$<http://grammar.example.com/politeness.gram#endPolite>;

$command = $action $object;
$action = /10/ open {TAG-CONTENT-1} | /2/ close {TAG-CONTENT-2} | /1/ delete {TAG-CONTENT-3} | /1/ move {TAG-CONTENT-4}; $object = [the | a] (window | file | menu);
```

Cross-Reference Examples

These two grammars illustrate referencing between grammars.

```
#ABNF 1.0 ISO-8859-1;

language en;
mode voice;
root $city_state;

public $city = Boston | Philadelphia | Fargo;

public $state = Florida | North Dakota | New York;

// References to local rules
// Artificial example allows "Boston, Florida!"

public $city_state = $city $state;
```

```
#ABNF 1.0 ISO-8859-1;

language en;
mode voice;

// Reference by URI syntax
public $flight = I want to fly to
$<http://www.example.com/places.gram#city>;

// Reference by URI syntax
public $exercise = I want to walk to
$<http://www.example.com/places.gram#state>;

// Implicit reference to root rule by URI
public $wet = I want to swim to
$<http://www.example.com/places.gram>;
```

Phone Number grammar

```
#ABNF 1.0;
mode voice;
language en-US;

// The lumenvox tag format tracks the current working draft of
// the W3Cs semantic interpretation proposal.
// 1.0 corresponds to the working draft released on 01 April 2003
tag-format <lumenvox/1.0>;

root $PhoneNumber;

/*
* ONE:"1" is shorthand for
* ONE {!{ $="1" }!}
* "$" refers to the current rule being matched ($Digit)
* So the net effect is that $Digit resolves to a one digit string
* after semantic interpretation.
*/
$Digit = (ONE:"1" |
          TWO:"2" |
          THREE:"3" |
          FOUR:"4" |
          FOUR:"5" |
          FIVE:"6" |
          SIX:"7" |
          EIGHT:"8" |
          NINE:"9" |
          (ZERO | O):"0" );|
```

```
/*
* $AreaCode resolves to a three digit string
* after semantic interpretation.
*/
$AreaCode = { $ = "" } ( $Digit { $ += $Digit } ) <3>;

/*
* $Number resolves to a seven digit string
* after semantic interpretation.
*
* $$ is shorthand for the last rule detected
* i.e. $Digit
*/

$Number = { $ = "" } ( $Digit { $ += $$ } ) <7>;

/*
* After semantic interpretation,
* $PhoneNumber resolves to a structure with two
member variable strings,
* areacode (which defaults to "858"), and number.
*/

$PhoneNumber = ([AREA CODE | ONE] $AreaCode
{ $.areacode = $$ } $Number { $.number = $$ } ) |
( $Number ) { $.areacode = "858"; $.number = $$ };
```

Chapter

7

Visual Dialplan Licensing

A Registration Code issued to you by Apstel, after the purchase, is required to use the Software in a non-trial mode.

Visual Dialplan Professional

When Visual Dialplan is registered on a personal computer, the registration code is tied to the personal computer Network Interface Card (NIC).
When the Asterisk Server is registered, the registration code is tied to the Asterisk Server definition.

The Visual Dialplan license allows software installation on two personal computers and registration of two Asterisk servers, one for development use and the other for production use.

In case you need to change or replace an already registered Asterisk server you can release the license associated with the registered Asterisk server and then reuse the same license to register the new Asterisk server (hardware replacement, Asterisk server replacement etc.). The server license release can be done easily with a single press of a button within Visual Dialplan.

In case you need to change or replace the desktop where you installed the software due to the hardware failures, upgrade to new hardware or similar, you can contact us via email and we'll release the license associated with the old/broken hardware so you can reuse the same license with your new hardware.

In case you need to manage more than two Asterisk servers or to use the software at more than two personal computers you may purchase an additional server or desktop license.

Under the terms and limitations of the License Agreement, Apstel grants you a nonexclusive, nontransferable license, without rights to sublicense, to:

- Make a number of copies of the Visual Dialplan less than or equal to the Number of Licensed Users for the purpose of installing a single copy of the Visual Dialplan on an equivalent number of personal computers, each of which is running the operating system for which the Visual Dialplan is designed
- Use the Registration Code to activate each copy of the Visual Dialplan to the extent permitted by your payment of applicable license fees under an Apstel approved licensing model
- Have up to the Number of Licensed Users use the Visual Dialplan (in object code form only)
- Make a number of Asterisk Servers less or equal to the Number of Licensed Asterisk Servers
- Use the documentation accompanying the Visual Dialplan in connection with permitted uses of the Visual Dialplan

Subject to the terms and limitations of this License Agreement, Apstel also grants you the following rights to:

- Re-register the Visual Dialplan on a personal computer after the change of the NIC, without acquiring the License Extension Agreement from Apstel, for one time only, unless otherwise stated on the previously received License Extension Agreement
- In addition to the Number of Licensed Asterisk Servers, to register one additional Asterisk Server that is to be used for development and testing purposes only, unless otherwise stated on the previously received License Extension Agreement

For the full license terms please refer to the LICENSE.TXT document located in your Visual Dialplan installation directory.

Visual Dialplan PBX Edition

When Visual Dialplan is registered, the registration code is tied to the PBX server Network Interface Card (NIC).

The Visual Dialplan license allows software installation on one (1) PBX server only.

In case you need to change or replace the PBX server where you installed the software due to the hardware failures, upgrade to new hardware or similar, you can contact us via email and we'll release the license associated with the old/broken hardware so you can reuse the same license with your new hardware.

Under the terms and limitations of the License Agreement, Apstel grants you a nonexclusive, nontransferable license, without rights to sublicense, to:

- Make backup copies of the software for the purpose of reinstalling the software in case of hardware failure, upgrade to new hardware or similar
- Re-register the software on a server after the change of the NIC
- Re-register the software on a server in case of the server hardware failure, upgrade to new hardware or similar
- Use the documentation accompanying the software in connection with permitted uses of the software
- Use the Registration Code to activate the software to the extent permitted by your payment of applicable license fees under an Apstel approved licensing model

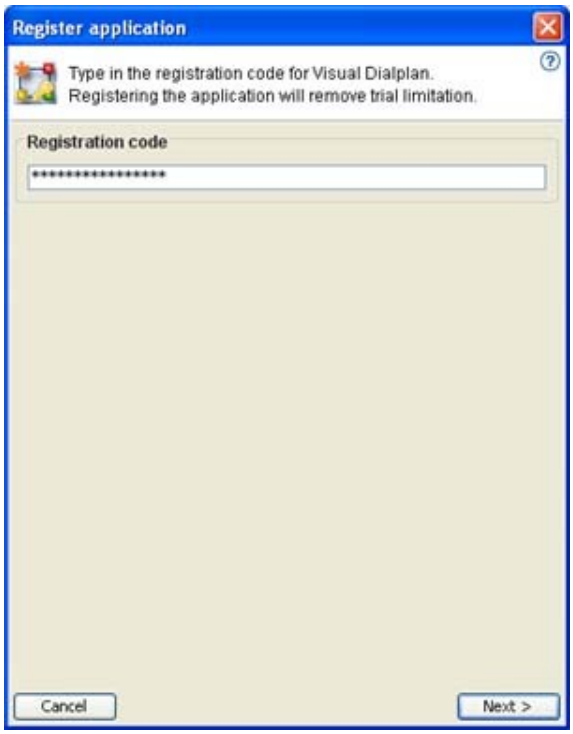
For the full license terms please refer to the LICENSE.TXT document.

Registration

Before you register the Visual Dialplan application read these instructions carefully. For license details, please refer to the Visual Dialplan License section.

Once you obtain the Registration Code by purchasing Visual Dialplan license, simply select Help/Register... from the main menu and follow the on screen instructions.

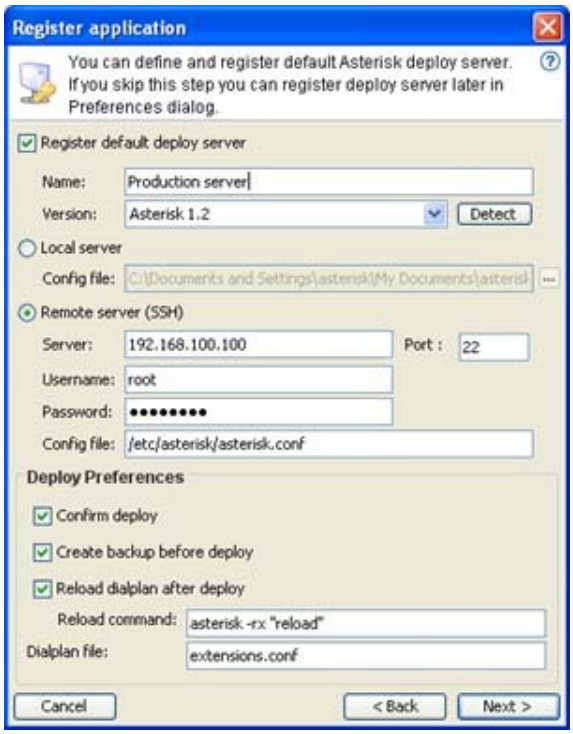
1. Enter registration code



Once you enter registration code click on the Next button at the bottom of the page. Note that Next button will be disabled until you enter valid registration code.

2. Define Asterisk deploy server

(Visual Dialplan PBX Edition does not have this step)



3. Confirm entered data and register

Important note: Make sure that your machine is properly connected to the Internet when registering the product. Visual Dialplan will contact the Apstel license server to complete the registration process.

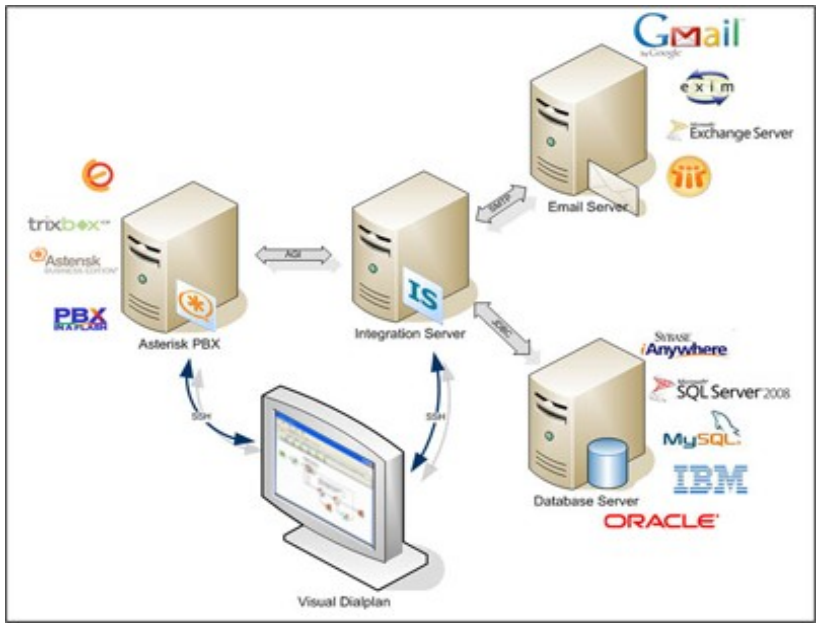
Chapter

8

Integration Server

The Integration Server (IS) is powerful application server that extends Asterisk dial plan functionality. It is specially designed to simplify working with third party servers (database servers, email servers, payment servers etc.) and services from within the Asterisk dial plan.

Integration Server comes with support for Visual Dialplan building blocks that provide intuitive interface to easily access those third party servers and service, like execute SQL queries on a remote database server (MS SQL, MySQL, Postgres etc.), send emails, process credit cards and more, directly from within the dial plan. It also extends Visual Dialplan with fully featured SQL query editor, email editor and payment editor.



Now you can easily test and execute SQL queries, send emails, process credit cards and do much more with Visual Dialplan.

How does it work - technical view

Integration Server (IS) is standalone server application that communicates with Asterisk server through AGI (Asterisk Gateway Interface) and acts as an AGI server that completes AGI requests initiated from Asterisk dial plan. These AGI calls can be simple database queries (database IS module) but also quite complex billing and credit card processing requests (additional IS modules are required) or other advanced functionalities depending on the modules loaded in the Integration Server.

An Asterisk Manager user is required in order to execute AGI calls.
Visual Dialplan automatically creates an Asterisk Manager user named is_user with a randomly generated password for this purpose. You can later modify both the Asterisk Manager username and password, if required.

Visual Dialplan deploys traditional extensions.conf code to the Asterisk server. In case the Integration Server functionality is required this code will contain AGI calls to Integration Server. At the same time, Visual Dialplan deploys resources to Integration Server required for those AGI calls.

When new call arrives at the Asterisk server the extensions.conf code (call flow) is executed and AGI request are sent to the IS. IS executes appropriate module (execute queries, send emails, process credit cards etc.) and returns resulting value and control back to the Asterisk dial plan.

How does it work - user view

Visual Dialplan comes with an Integration Server view, this is where the connection to the IS should be defined. Open this view and define a connection to the IS. Once the IS node is created, you can expand it and define connection to other servers and services supported by IS (DB server for example). Now you are ready to use the appropriate building blocks to call that particular IS functionality from within the dial plan (DB building block from IS sheet in this case). It is that simple.

Integration Server installation

Integration server can be installed on Microsoft Windows or Linux operating system.
It can be installed on the same box where Asterisk is installed or on a separate server.

Integration Server web interface

Integration Server comes with integrated web interface. It is the main interface to set up Integration Server access parameters, to maintain the server and manage the licenses.

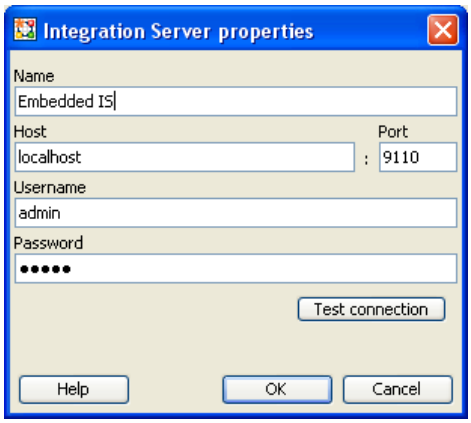
Default parameters to access Integration Server are:
Username: **admin**
Password: **admin**
Web interface port: **9111**

NOTE: For the security purposes it is our strong recommendation to change the web interface username, password and port immediately after the server installation.

Integration Server properties

Integration Server can be accessed via HTTP and directly from Visual Dialplan.

Username and password to access IS are the same in both cases.



Field	Description
Name	Internal name for the Integration Server that will be used to reference this particular IS.
Host	The URL or IP of the server where Integration Server is installed.
Port	Port to access Integration Server. Default value is **9110**.
Username	Integration Server user name. Default value is **admin**.
Password	Integration Server password. Default value is **admin**.

Note:
It is our strong recommendation that you change the default username, password and port values immediately after the server installation.

Default values can be changed using the Integration Server administrative web interface which can be accessed at the following URL:

http://<Integration Server Host>:9111

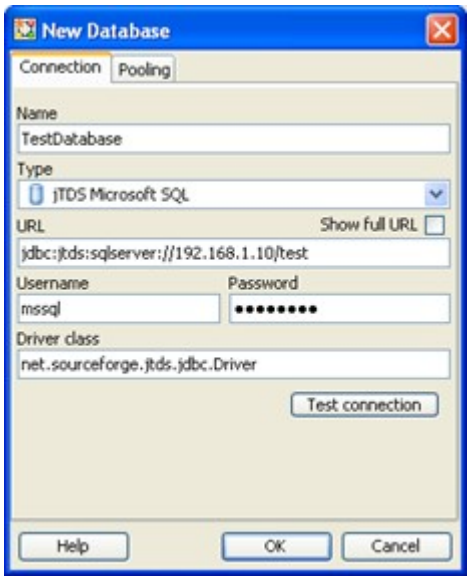
Database module

Executes SQL queries from the dial plan.

This module is designed to simplify SQL query execution from within the dial plan against any database that provides JDBC drivers. The module will handle connection to the database server, execution of SQL query and management of the result set.

To create connection to new database server simply right click on the selected *Database resource* node and choose *New database* from the drop down menu.

Database connection properties



Field	Description
Name	Database connection name.
Type	Database server type. Several database types are supported out of the box while other database types can be supported by adding the JDBC drivers provided by database vendor. Defines the type of the database. Visual Dialplan will list only those databases for which a JDBC driver is available. The following database types will be available out of the box: • MySQL • Sybase • Microsoft SQL • Postgres SQL • JDBC ODBC bridge • Custom database Custom database drive that can be used to install any other native database driver.
URL	Database connection URL (check particular database server documentation for more

	information). Integration Server provides default URL for each supported database type. Make sure to replace parameters under the '<>' with adequate values (hostname, dbname, alias etc.).
Username	Database username used to connect to the database.
Password	Database password used to connect to the database.
Driver class	Driver that will be used by Integration Server to work with particular database type.

Note: Integration Server and connection to database server should be configured properly in order to use DbQuery component. Make sure to set database remote access privileges properly in order to access database server from Integration server

Integration Server resources (database connection, SQL queries etc.) are stored in the VDP file together with other call flow related information.

Field	Description
Connection pool	Initial Size - The initial number of connections that are created when the pool is started. Max Active - The maximum number of active connections that can be allocated from this pool at the same time, or -1 for no limit. Max idle - The maximum number of connections that can remain idle in the pool, without extra ones being released, or -1 for no limit. Min idle - The minimum number of connections that can remain idle in the pool, without extra ones being

	created, or zero to create none. Max wait - The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception, or -1 to wait indefinitely.
Transactions	Read only - The read-only state of this connection created by this pool. Transaction isolation - The default TransactionIsolation state of connections created by this pool.
Validation	Validation query - The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query MUST be an SQL SELECT statement that returns at least one row. Test on borrow - Indication of whether connections will be validated before being borrowed from the pool. The Validation Query must be set. Test on return - Indication of whether connections will be validated before being returned to the pool. The Validation Query must be set.

Database pooling parameters shouldn't be modified unless you really know what you are doing.

Adding additional database drivers

Integration server and Visual Dialplan are shipped with JDBC drivers for the following databases:

- MySQL
- Sybase
- Microsoft SQL
- Postgres SQL
- JDBC ODBC bridge

Those drivers are stored in the /jdbc subfolder in the Integration Server and Visual Dialplan installation folders.

If you need to access a database for which Integration Server and Visual Dialplan does not have JDBC drivers for, you will have to copy JDBC drivers for your database in the /jdbc subfolder in the Integration Server and Visual Dialplan installation folders. Both Integration Server and Visual Dialplan need to be restarted after the drivers are copied in the /jdbc subfolder.

Note:

Some of the out of the box JDBC drivers shipped with Integration Server and Visual Dialplan are open source versions and are not recommended for production systems. It is recommended to replace them with the JDBC drivers provided by your database vendor.

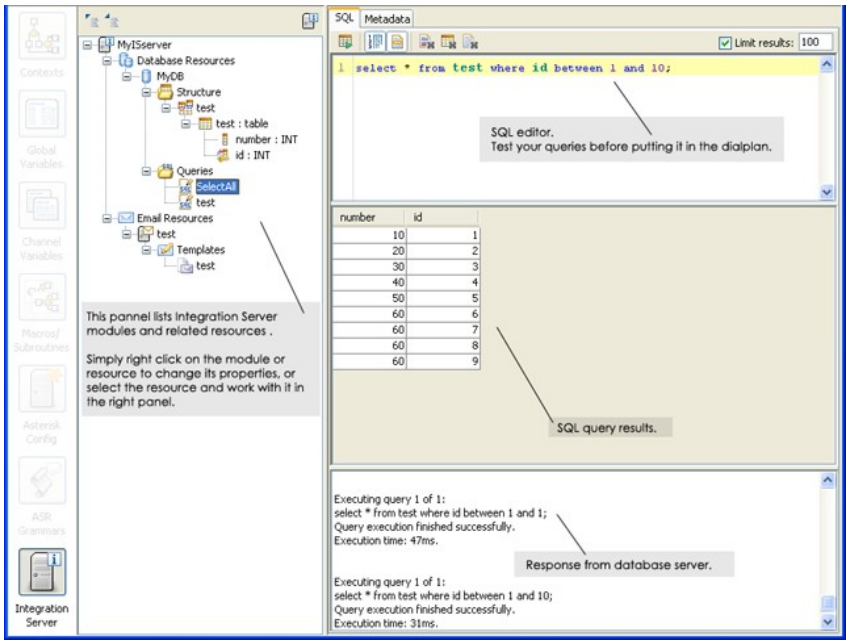
Embedded database

Integration Server is shipped with embedded HSQL database (http://hsqldb.org/). To create a connection to the embedded database select **IS Embedded Database** from the Type drop down in the Database Properties panel.
Default port that this embedded database listens on for the JDBC connections is **9112**. The name of the database is **isdb**.
Username and password used to access the embedded database are the same as the ones for Integration Server; **admin**/**admin** by default.

Database module view

The Database module view consists of the following panels:

- SQL query editor
- SQL query result panel
- Database response panel
- Metadata panel

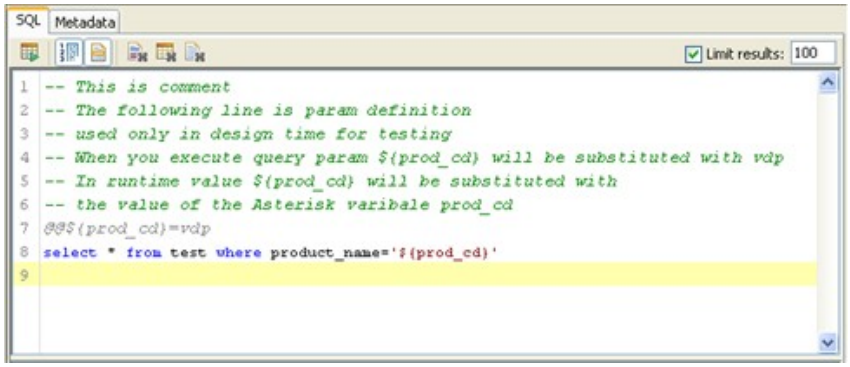


SQL query editor

This is the place to type and test SQL queries before using them in the dial plan.
Simply type in the SQL query, click on the Execute query button (ctrl+Enter) and the query will be executed.
The result set will be displayed as well as the database server log records related to this query.

You can also use parameters (dial plan variables) within SQL query. Simply define the parameter and use it in the query. The parameter will be used at design time only and replaced with Asterisk variable value at run time. Here is an example.

```
-- This is the comment.
-- The following line is parameter definition used only in the design time.
@@${ID}=123
-- When you execute the query in the SQL query editor
-- the parameter ${ID} will be replaced with number 123.
-- In runtime ${ID} will be replaced with Asterisk variable ${ID} value.
select * from lic_info where product_id = '${ID}'
```



You can also limit the maximum number of returned rows (default value is 100).

SQL query result panel

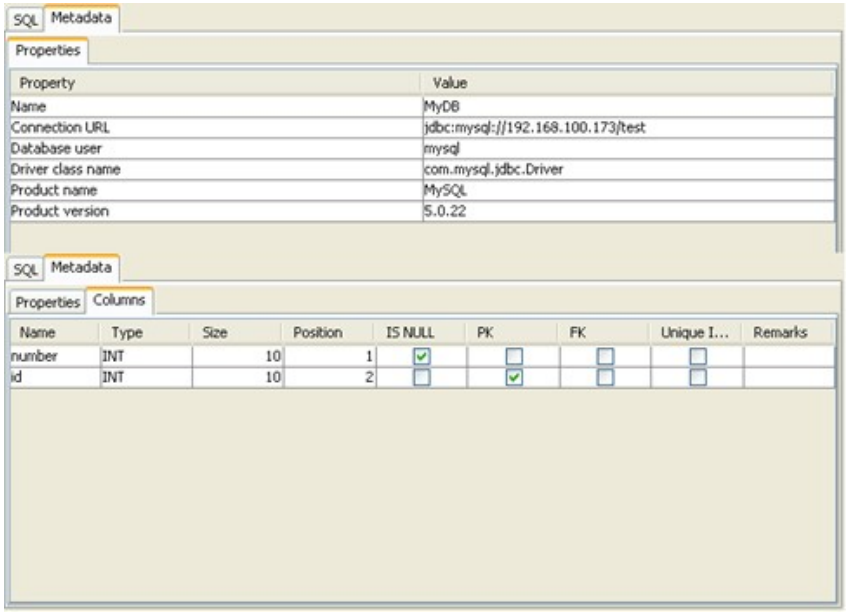
Returned result set will be displayed here.

Database response panel

Database server log records related to executed query will be displayed here.

Metadata panel

Under this panel you can check the database connection parameters like Connection URL, database user name, database server version, table names, column names and properties and similar.

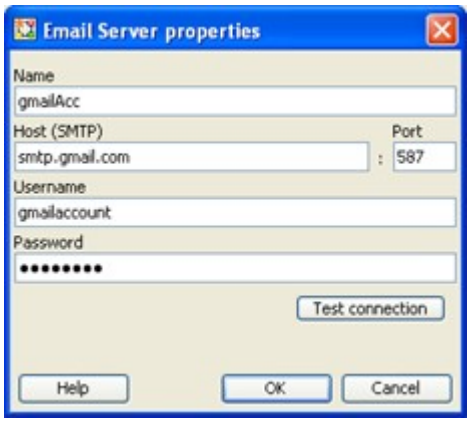


Email module

Send emails from the dial plan

This module is designed to simplify email sending from within the dial plan. The module will handle connection to the SMTP server and send plain text or HTML emails.

To create a connection to the SMTP server simply right click on the selected *Email resource node* and choose *New email server* from the drop down menu.



Field	Description
Name	Email server connection name.
Host (SMTP)	URL or the SMTP server (for example smtp.gmail.com).
Port	The port that SMTP server declares for email clients (the same port that your email client use). For plain text authentication it is usually port number 25 but it may vary depending on the authentication type and SMTP server. For example, gmail usually uses port number 587.
Username	Email account username (the username you type to login to your email account).
Password	Email account password (the password you type to login to your email account).

Note:

Integration Server and connection to SMTP server should be configured properly in order to use SendEmail component

Embedded email server

Integration Server is shipped with embedded JES email server (http://www.ericdaugherty.com/java/mailserver/). To create a connection to the embedded email server use the port **9113**.
Username and password used to access the embedded SMTP email server are the same as the ones for Integration Server; **admin**/**admin** by default.

Email module view

This is the place where email templates should be defined. Once the email template is defined it can be used to send emails from the dial plan using SendEmail component.

All dial plan variables used in the email template, ${variable}, will be replaced with its corresponding values at runtime.

For example, the HTML email template, with *name* variable, and a predefined value of *Michael* , for the email preview/test purpose only (at runtime the variable *name* will be replaced with the value of the *name* dial plan channel variable), an will look like this:

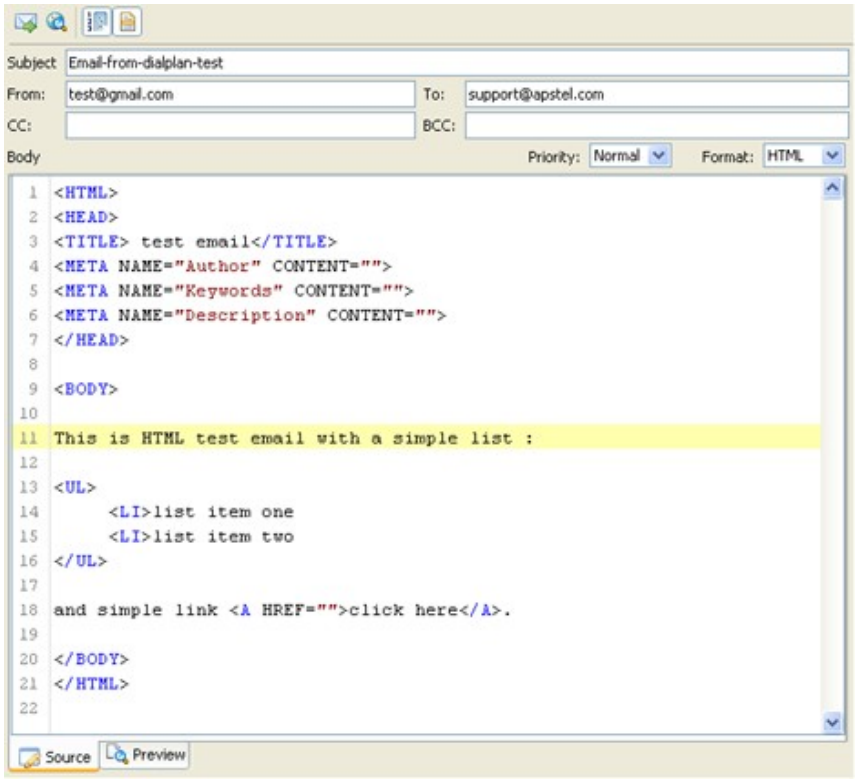
<!@@ ${name}=Michael>

<h1>Test Mail</h1>

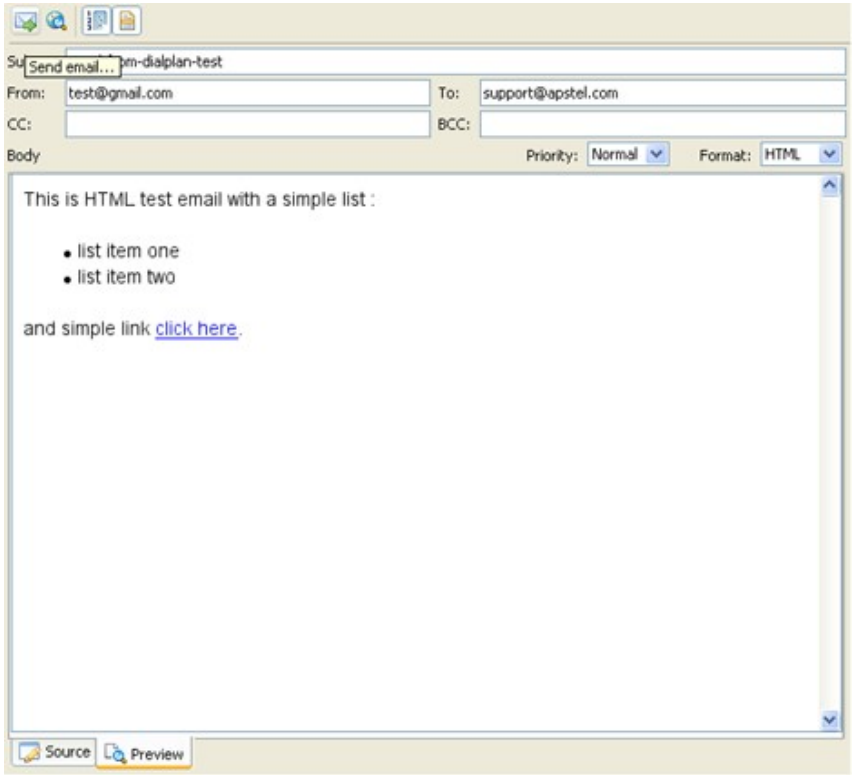
Hello ${name},

Thank you

Besides standard email parameters like Subject, From, To, CC and BCC fields there is also ability to set email priority (Low, Normal, High) and email type (HTML or plain text).



Before using the email template make sure to preview it in Visual Dialplan using the Preview panel, and preview it in your default web browser by clicking on the Preview in browser button, or send a test email by manually clicking on the Send email button.

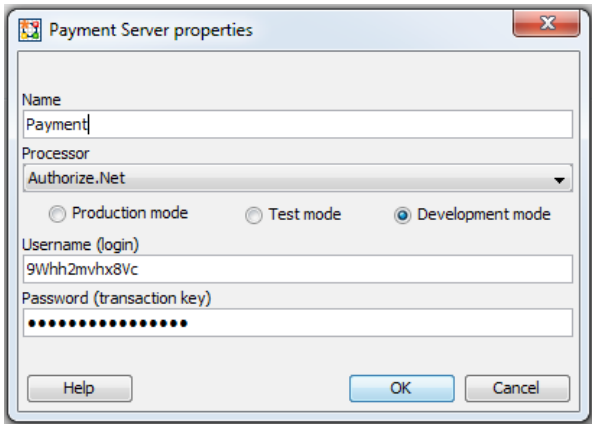


Payment module

Process payments from the dial plan

This module is designed to process credit card payments from within the dial plan. The module will handle connection to the payment server and will process payments.

To create a connection to the payment server simply right click on the selected *Payment resource* node and choose New payment server from the drop down menu.



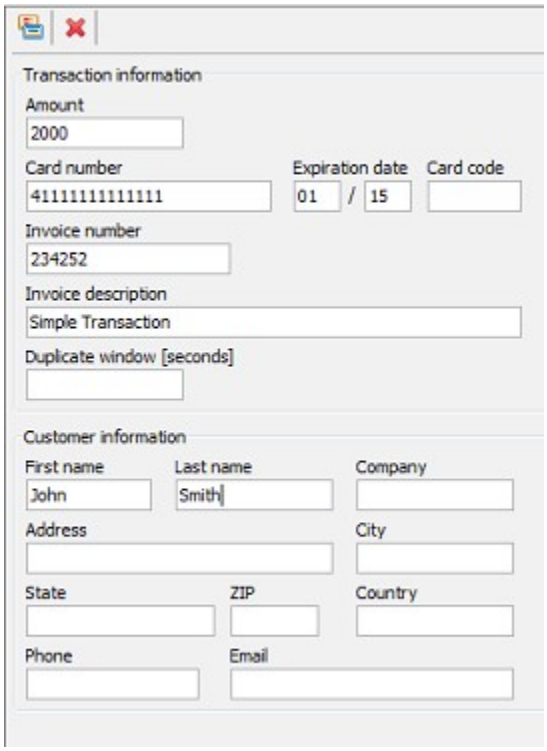
Field	Description
Name	Payment server connection name.
Processor	Choose credit card processor (for example Authorize.NET). Pick one of the following options: 1, Production mode 2. Test mode 3. Development mode
Username	Merchant login ID.
Password	Merchant transaction key.

Note:

Integration Server and connection to Payment server should be configured properly in order to use ProcessPayment component.

Payment module view

This is the place where payment transaction should be tested.



Click on the "send payment" button to initiate payment transaction.

Integration Server Licensing

Integration Server (IS) is licensed per channel (simultaneous call) per IS module, and comes bundled with several free licenses:

- 1 free channel license for database module (one simultaneous call with database query)
- 1 free channel license for email module (one simultaneous call with email initiation)
- 1 free channel license for payment module (one simultaneous call with payment initiation)

This means you can test and develop database, email and payment driven Asterisk dial plans for free, without purchasing an additional Integration Server license.

In case you anticipate more than one simultaneous call with database query, email or payment initiation you may consider purchasing additional channel licenses. For example, if you anticipate five simultaneous calls with database query and three simultaneous calls with email initiation (send emails from the dial plan) you should purchase 5 - 1 = 4 channel licenses for database module and 3 - 1 = 2 channel licenses for email module.

Integration Server licenses can be purchased directly from the Integration Server web interface on the Licenses page. A Registration Code issued to you by Apstel, after purchasing, should be entered on Licenses page (IS web interface) in order to activate channel license(s).

When the Integration Server is registered on a computer, the registration code is tied to the computer Network Interface Card (NIC).
The Integration Server license allows software installation on one computer only.

In case you need to change or replace the computer where you installed the software due to the hardware failures, upgrade to new hardware or similar, you can contact us via email (support@apstel.com) and we'll release the license associated with old/broken hardware so you can reuse the same license with your new hardware.

In case you need to run more than one Integration Server instance you would need to purchase additional licenses.

Under the terms and limitations of the License Agreement, Apstel grants you a nonexclusive, nontransferable license, without rights to sublicense, to:

- Make backup copies of the Integration Server for the purpose of reinstalling Integration Sarver in case of hardware failure, upgrade to new hardware or similar
- Re-register the Integration Server on a computer after the change of the NIC
- Re-register the Integration Server on a computer in case of the computer hardware failure, upgrade to new hardware or similar
- Use the Registration Code to activate channel licenses to the extent permitted by your payment of applicable license fees under an Apstel approved licensing model
- Use the documentation accompanying the Integration Server in connection with permitted uses of the Integration Server

For the full license description please refer to the LICENSE.TXT document located in the application installation directory.

Index