WEB BASED FRAMEWORK FOR 911 EMERGENCY CALL HANDLING USING OPEN SOURCE ELASTIX PBX

Shimoga Prasad Shashi Kiran

Master of Science in Internetworking

UNIVERSITY OF ALBERTA

2011-12

Abstract

This document provides a brief overview on the Analysis and prototype development of Emergency call handling module in Elastix web server GUI. It aims at providing a front-end web framework with necessary scalability features for complete solution development resulting in a full-fledged application that can be integrated with any Elastix server package.

Previous developer in the earlier phase of the 911 Emergency Call handling project has devised an algorithm which will pick an available telephone line to free the resources so any user in emergency can dial 911 in a congested environment of the PBX Elastix. This project aims on integrating the developed algorithm with web based solution framework using Elastix. This module will enable any novice user with sufficient access level privileges to define and understand the context and priorities from the web based GUI. The cost priorities define the order in which channels get freed when emergency calls need to be handled in the PBX system.

The following is the brief activity list which focuses on Analysis work done as part of 911 Emergency call handling application development:

- Development of new web GUI framework that can be integrated with Elastix
- Integration of Java script(Developed Algorithm) with web based framework
- Packaging the functional web based framework with Elastix
- Testing Elastix web based GUI framework with necessary and additional functionalities for 911 emergency call handling and congestion control

This Capstone project report is submitted as part of MINT 709 requirement and this will act as a suitable reference for all the developers and testers interested in exploring Open source PBX environment with Asterisk-Java and Asterisk-PHP scripting and doing new developments in Elastix VOIP Communications Server.

Acknowledgements

I would like to thank Dr.MacGregor for giving me an opportunity to work on the area of Open source PBX Scripting. I would like to extend my thanks to Mr.Shahnawaz Mir for supervising the project and guiding in the right direction. Finally I would like to convey my special thanks for Mr. Kirupakaran Pirasath, who was the developer in the first phase of the project and whose initial contribution and research in Asterisk PBX helped in exploring different facets of the Elastix PBX and thus paved way for delivering a better enhanced quality solution through Elastix GUI in the field of VoIP telephony.

Table of Contents

List of diagrams and tables used in the report:

List of Abbreviations and Acronyms used in this Report:

AGI	- Asterisk Gateway Interface
Asterisk-Java	- Asterisk for Java
AGI-PHP	- Asterisk Gateway Interface in PHP
AMI	- Asterisk Manager Interface
ASM	- Asterisk Service Manager
API	- Application Programming Interface
CEO	- Chief Executive Officer
CLI	- Command Line Interface
DTMF	- Dial-Tone Multi-Frequency
DAHDI	- Digium Asterisk Hardware Device Interface
GUI	- Graphical User Interface
IP	- Internet Protocol
IVR	- Interactive Voice Response
OS	- Operating System
PBX	- Private Branch Exchange
PC	- Personal Computer
PHP	- Hypertext Preprocessor
POTS	- Plain Old Telephone System
PSAP	- Public Safety Answering Point
PSTN	- Public Switched Telephone Network
SIP	- Session Initiation Protocol
TDM	- Time Division Multiplexing
UML	- Unified Modeling Language
MVC	- Model View Controller
YATE	- Yet Another Telephony Engine

VoIP	- Voice over Internet Protocol
HTML	- Hypertext markup language
AstPHPAGI	- Asterisk PHP-AGI Library

Introduction

Elastix, an open source VOIP communications server provides an ideal platform for communication and telephony systems to utilize the latest technologies in the VOIP Industry and leverage the maximum benefits out of it. The organizations can make their telephony environment more effective and productive with the excellent usage of Elastix and its customization with different communication systems.

The Elastix server runs on CentOS and it includes a web server component by which the system is configured via the Elastix GUI.Elastix is an open source software used to provide all required functionalities of a VOIP Communications Server. Elastix provides industry standard telephony system with various features like Voicemail, call-recording, Conference, Monitoring and many more features to the user in the form of a simple GUI framework. A user with sufficient login access to Elastix server can explore the various functionalities that Elastix is offering in the open source PBX environment.

One of the greatest benefits of using the Elastix web server and its GUI is its open source and unified nature in telephony systems. The Elastix GUI acts as a single point of access by which the entire VOIP communication server and its various channels can be suitably modified and changed according to the requirements as and when the changes are necessary. It helps organizations to substantially manage costs by deploying and maintaining the VOIP communication servers in their organization environment that are highly flexible and stable.

One of the important features of Elastix is its flexibility to incorporate additional design features in its architecture. Majority of the users do not understand the changes that happen beyond the Elastix web server and this iterates the need for additional features to be added into the Elastix Web GUI so that novice users will have better visibility and control in handling the telephony operations from the web GUI.

The 911 Emergency calls are normally routed to Public Safety Answering points (PSAP) across different locations in Canada and are given highest importance in terms of dialing and answering them. Hence, it provides a great opportunity for the software developers and researchers to come up and deliver new additional exceptional emergency dialing services that will benefit the organizations in the long run.

2. Elastix Asterisk Overview and Functional Description:

The below section provides detailed information of the Elastix Web GUI and its interaction with the Asterisk.

2.1Elastix Web GUI Overview and Terminologies:

2.1.1PBX Configuration in Elastix:

The Elastix Web GUI provides a simple feature to create extensions and different groups to configure PBX .The context and priorities are defined for each extension which will be saved in the database of the Elastix web server and will be acted upon when calls from those specific groups are made from it.

2.1.2The Asterisk and its usage:

Asterisk is the software that helps VoIP gateways, conference servers and other PBX tools in establishing a collaborative and supportive environment for unified VoIP communication.

The asterisk gateway interface (AGI) is a standard interface by which asterisk or Elastix asterisk dial plan may be monitored and maintained by the external programs that will run on those servers. The Asterisk dial plans are suitably written in such a way that the AGI(Asterisk Gateway Interface) will act as an gateway for scripts or programs that define advanced logic and perform some actions and operations based on the commands executed in the Asterisk.

The AGI scripts communicate over the Asterisk defined specific communication channels and Asterisk sends lists of its variables to macro-dialouts after the AGI script begins its execution.

2.1.3Asterisk and PHP scripting:

The AGI scripts can be written in PHP with suitable modifications but the basic rules of AGI scripting will still apply to the program that will be written in PHP .There is also the support of PHPAGI Library for writing advanced AGI Scripts in PHP.

The Elastix web server code runs in PHP and writing AGI scripts in PHP provides an added advantage to the programmer with the ability to modify and change not only the AGI scripting code but also the PHP code that makes up the web based Elastix GUI.

2.2 Previous Research and Work done on this topic:

The developer in the previous phase of the project has devised an algorithm that has the following features:

- The user groups will be assigned priorities with specific costs and the user groups are classified as Student, Visitor, CEO based on the business process model and hierarchy of the organization
- The priorities with costs will be assigned for all calls made outside of the organization and especially under the circumstances of handling emergency calls and the incoming calls coming through PSTN will not be assigned priority.

• The non-emergency (non-911) lines will be disconnected based on the calls having the lowest priority in terms of cost assigned to them in macro-dial out trunks

911 Emergency calls:

- 911 Emergency calls will all have the highest priority compared to any other non-emergency calls.
- There is a global counter called EMERGENCY that will get incremented each time the 911 call is connected to the PSAP
- There will be no disconnection of 911 emergency calls at any point of time.
- The users will try to attempt to dial 911 or non 911 will hear congestion tone or IVR when all outgoing PSTN lines are in use with Emergency handling application
- When each call is hung up after using 911 emergency application, the Global Counter EMERGENCY will be decremented by 1

The previous developer had proposed and developed the solution that consisted of checking whether all the four lines/trunks were busy dialing some calls. Based on that decision, then further decision was made on whether they were busy dialing the non-911 calls or 911 calls. If they were dialing non-911 calls and if there is emergency call to attend then it would pick the line which is attending non-911 call and which is present at the lowest level in terms of priorities being assigned to them .

The Emergency call which needs to be attended would then be assigned the highest priority and it would be connected. If all the available lines were not busy, then they would directly assign the emergency 911 calls .If all the calls were busy with 911 dialing, then an IVR message would be delivered to the caller informing him that the line was congested with emergency calls and they are not in a position to disconnect any of the emergency calls and would need to wait until one of the lines become available.

2.3 Need for Elastix GUI Development for Emergency Call handling application:

The previous developer had devised the algorithm and the code was written in java. The main priority was to develop an Emergency call handling application using Java/PHP and Asterisk scripting. The application in first phase of the project had certain limitations which are listed below:

a) The Novice user was unable to identify the application since the application was running in the Asterisk and the only way to identify the code was to login to the Asterisk CLI in Elastix and run the program.
b) This was tedious for a novice user since most of the users will probably have less knowledge on programming languages and will find it difficult to understand the application
c) The finer aspects of the application are complex for a user to understand and make it operative.
d) The development of Elastix GUI for Emergency call handling application was a typical user requirement since the users prefer to use the application in the simplest possible

manner and they ideally like to have all complex features to be hidden encapsulated from them.

e) They had to manually run the script that handles the Emergency call handling application.

The following are the advantages of using web based Elastix GUI:

- The creation of web based application always results in higher productivity for organizations.
- The users will have less load of performing tasks as web GUI does most of the work and they do not have to remember each and every important task when they use the application
- It requires less experience, less training for non-technical people to understand and use the application.
- It nullifies the users need to adapt to usage of complex command languages in Asterisk scripting.
- The Elastix GUI do not require large amount of memory or processing power to run the Emergency call handling application.

2.4 Proposed Solution for web-based framework in Elastix GUI:

Based on the requirements, there were two possible solutions for development of web based framework for Elastix:

1) ***Conversion of entire existing application code in java to PHP and development of new web interface in PHP for Elastix GUI***.

Significant analysis was done on studying the feasibility of converting the application code written in java to PHP since the Elastix web interface code is predominantly in PHP. But the Analysis results showed that this option had lot more disadvantages than advantages. The following are the list of disadvantages of using the above option:

a) The existing application code had written in Java with the support of Asterisk Java scripting. Conversion to PHP meant that entire code had to be rewritten with Asterisk PHP scripting. Further analysis resulted in findings that Asterisk java scripting has greater degree of support with Java-AGI libraries when compared to PHP AGI libraries in terms of their integration with Asterisk AGI.

b) PHP developer modules in Elastix offer very little flexibility in terms of writing a completely different PHP code that does not align with the current web based Elastix GUI.

c) The developer module will not allow the developer to load and integrate different PHP programs that does not follow the basic structure of packages from which Elastix GUI was developed i.e. changes to PHP code can only be made once it is loaded into the Elastix GUI with very little flexibility in changing the code as and when required.

d) The application java code is embedded inside the AGI scripting and it gets called inside the AGI script without any technical difficulties where as PHP scripts inside AGI script under current scenario tends to impose technical limitations
e) The current Elastix code makes writing new PHP code more complex without the usage of external libraries like php-agi or AstPHPAGI

2) ***Maintaining the existing code in Java and development of new web interface in PHP and their integration with the help of AGI Scripting in addition of the usage of PHP-Java bridge for future PHP-Java embedded extension code to be written in Emergency call handling application***

The above solution had additional advantages and offers more flexibility in terms of expanding the Emergency call handling application. The important feature to be noted here is additional support that the AGI scripting provides for java. The only drawback we can find in this solution is the additional installation of java files and mapping it into the Asterisk. But once the required java files are installed and required java path is set up then it offers unparalleled support to Asterisk.

The current Elastix environment also enables the usage of open source PHP-Java bridge , when installed in the Elastix server offers flexibility of embedding the java classes inside the PHP code .This is where its efficient usage comes into use when Elastix poses certain technical limitations of embedding the PHP code within the AGI script, the installed Java-PHP bridge helps in offering the additional support of embedding the Java classes inside the PHP code and overcomes those limitations.

3. Elastix Server GUI Design for Emergency Call Handling Application:

3.1 Design Specifications:

The Elastix web framework design was completely based on web development in the Elastix web server and the overall objective was to ensure additional GUI capabilities are embedded in Elastix for the Emergency call handling application. Accordingly, since it is web development based framework and Elastix GUI is already developed in PHP, PHP was chosen as one of the programming languages from which the additional features of the Elastix GUI would be developed.

Hardware and Software Specifications:

The Elastix with CentOS server present in the MINT Lab was chosen for the Elastix GUI web development .The Asterisk server with Asterisk 1.6.2.20 version was used to design new features in Elastix GUI. The following specifications formed the basic Elastix environment with which the development of web features in Elastix GUI was carried out:

- A PC/Notebook with at least 512MB/20GB/1 core
- Elastix 1.6 (bundled with CentOS release 5.7 (Final) and Asterisk 1.6.2.20)
- TDM card - Wildcard TDM410P Board 1
- Yate VoIP SIP clients(soft-phones)
- PHP 5.1.6 (cli) Zend Engine v2.1.0
- Java version "1.6.0_23"
  Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
  Java HotSpot(TM) 64-Bit Server VM (build 19.0-b09, mixed mode)

  Elastix GUI development is the ideal way to build custom telephony solutions and applications in Asterisk and the best way to establish communication between telephony systems through Asterisk is to adopt SIP protocol for trunking and other configurations. The user groups communicate from the Yate VoIP clients (soft-phones) with different extensions by adopting Session Initiation Protocol (SIP).

  The following flowchart demonstrates the diagrammatic representation of the proposed web based Elastix GUI solution:

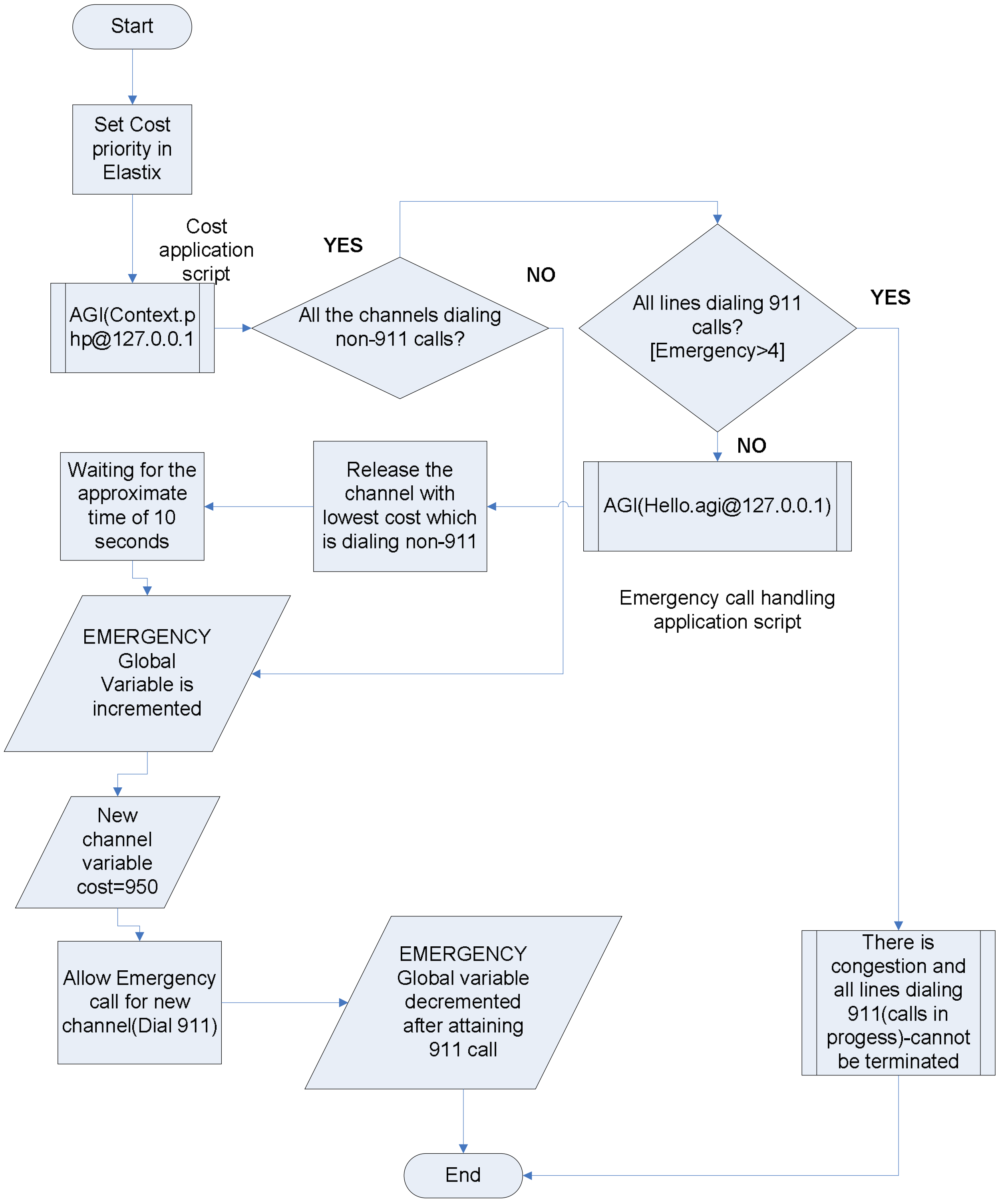

Start
Set Cost priority in Elastix
Cost application script
AGI(Context.php@127.0.0.1
YES
All the channels dialing non-911 calls?
NO
All lines dialing 911 calls? [Emergency>4]
YES
NO
Waiting for the approximate time of 10 seconds
Release the channel with lowest cost which is dialing non-911
AGI(Hello.agi@127.0.0.1)
Emergency call handling application script
EMERGENCY Global Variable is incremented
New channel variable cost=950
Allow Emergency call for new channel(Dial 911)
EMERGENCY Global variable decremented after attaining 911 call
There is congestion and all lines dialing 911(calls in progess)-cannot be terminated
End

Fig 3.1.1 : Flowchart of the Emergency call handling application

Design of Emergency call handling application and how it aligns into the purview of Elastix web framework:

This section explains on how the base algorithm developed by the developer in the previous phase of this project is integrated with Elastix web framework and how the users can see the functionality of Emergency call handling application through Elastix GUI.

The following extension numbers assigned in the base algorithm of previous developer is carried forward:

Name of the Extension	Extension numbers
CEO	300
Manager	400
Staff	500
Student	600
Visitor	700

Fig 3.1.2: Table about the extensions created for user groups

This user groups will have the same structure below in extensions.conf as defined in the base algorithm:

```
[CEO]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
[Manager]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
[Student]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
[Staff]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
[Visitor]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
```

The following emergency module gets acted upon and the AGI script runs in the background:

```
[emergency]

exten => s,1,ExecIf($[${EMERGENCY}>=4],Congestion)

exten => s,n,Set(GLOBAL(EMERGENCY)=$[${EMERGENCY} + 1])

exten => s,n,Set(TEST=911)

exten => s,n(dial),Dial(${EMERGENCY_TRUNK}/${EMERGENCY_NUM})
exten => s,n,GotoIf($["${DIALSTATUS}"="CONGESTION"]?agi)

exten => s,n,Wait(12)

exten => s,n(agi),Agi(agi://localhost/hello.agi)

exten => s,n,Goto(dial)

exten => h,1,ExecIf($[${TEST}=911],Set,GLOBAL(EMERGENCY)=$[${EMERGENCY} - 1])
```

The base algorithm is embedded inside the script hello.agi and this gets called only when the Emergency call handling situation occurs.

The contexts that are set in the emergency will always refer from the macro-dialout trunk defined in the extensions_additional.conf file in /etc/asterisk folder of the Elastix server.

```
[macro-dialout-trunk]

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"],Set,COST=${Cost})

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Manager"],Set,COST=${Cost}-100}

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Staff"],Set,COST=${Cost}-200)

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Student"],Set,COST=${Cost}-300)

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Visitor"],Set,COST=${Cost}-400)
```

When 911 calls are made from the outside world, priorities are set for the user-groups using the $Cost variable defined in the AGI script that gets called in from Elastix GUI to check the existing values and assigns the values from the local variables present in the script to global variables present in the macro-dial out trunk of the dial plan in Asterisk

When all non-emergency calls are in progress and there is an emergency call from one of the user groups, it will trigger the Emergency process and an asterisk channel has to be freed which is kept track by global variable EMERGENCY which indicates congestion in the line and caller group which has the lowest cost priority gets dropped eventually.

So the work flow from the Elastix GUI is defined as follows:

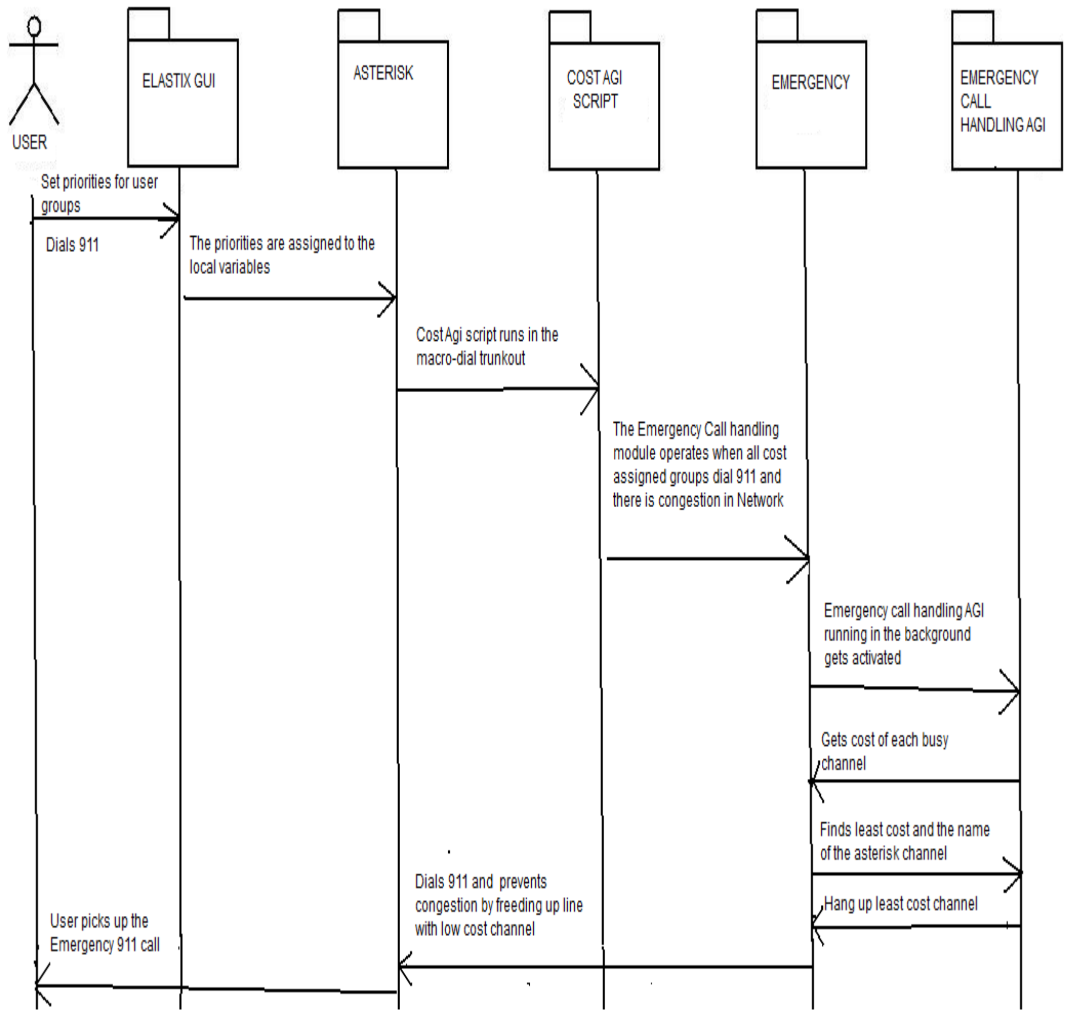

USER
ELASTIX GUI
ASTERISK
COST AGI SCRIPT
EMERGENCY
EMERGENCY CALL HANDLING AGI
Set priorities for user groups
Dials 911
The priorities are assigned to the local variables
Cost Agi script runs in the macro-dial trunkout
The Emergency Call handling module operates when all cost assigned groups dial 911 and there is congestion in Network
Emergency call handling AGI running in the background gets activated
Gets cost of each busy channel
Finds least cost and the name of the asterisk channel
Hang up least cost channel
Dials 911 and prevents congestion by freeding up line with low cost channel
User picks up the Emergency 911 call

Fig 3.1.2: Work flow processes in the Emergency Call Handling application

3.2 Design and Development of Web Framework in Elastix:

This section provides a brief overview on methodologies followed in designing the new emergency call handling framework in Elastix.

3.2.1Installation and Configuration of PHP-Java Bridge:

The previous developer has devised the algorithm in Java using the Asterisk-Java scripting. The entire module structure of the Elastix is PHP oriented. This necessitated the need to integrate PHP and Java which are two different powerful programming languages existing in the world today.

This section explains the installation and configuration work carried out to implement the open source Java-PHP bridge:

```
[root@elastix ~]# which java

/usr/bin/java

[root@elastix ~]# ll /usr/bin/java

lrwxrwxrwx 1 root root 26 Jan 14  2011 /usr/bin/java -> /usr/java/default/bin/java

[root@elastix ~]# export JAVA_HOME=/usr/java/default/bin/java
```

If Java is not installed in the Elastix System, it can be installed using following command:

```
[root@elastix ~]# yum install java
```

Java can also be installed manually on the Elastix server.

Need to check if Java is currently installed in the system:

```
[root@elastix ~]# java -version

java version "1.6.0_23"

Java(TM) SE Runtime Environment (build 1.6.0_23-b05)

Java HotSpot(TM) 64-Bit Server VM (build 19.0-b09, mixed mode)
```

Installation of Open-source PHP Java bridge:

```
[root@elastix ~]#rpm2cpio php-java-bridge-4.1.8-1.FC5.i386.rpm

[root@elastix ~]#cpio –ivd

[root@elastix ~]#tar xzf php-java-bridge-4.1.8.tar.gz

[root@elastix ~]#cd php-java-bridge-4.1.8

[root@elastix ~]#phpize
```

(Or)

```
[root@elastix ~]#rpm2cpio php-java-bridge-4.1.8-1

[root@elastix ~]#./configure –with-java=$JAVA_HOME && make

[root@elastix ~]#./configure –prefix=/usr/src/php-java-bridge-4.1.8 –with-java=$JAVA_HOME
```

Add the following lines in the /etc/php.ini folder :

```
;Java Extensions;

extension=java.so
```

```
[java]

java.java_home="/usr/java/default/bin"

java.java="/usr/java/default/bin/java"

java.class.path="/usr/src/usr/lib/php/modules/JavaBridge.jar"

java.library="/usr/java/jdk1.6.0_23/jre/lib/amd64/server/libjvm.so"

java.library.path="/usr/src/usr/lib/php/modules"

java.log_level=3
```

Once the PHP-Java bridge is installed, the PHP programs can be created and executed using Java classes.

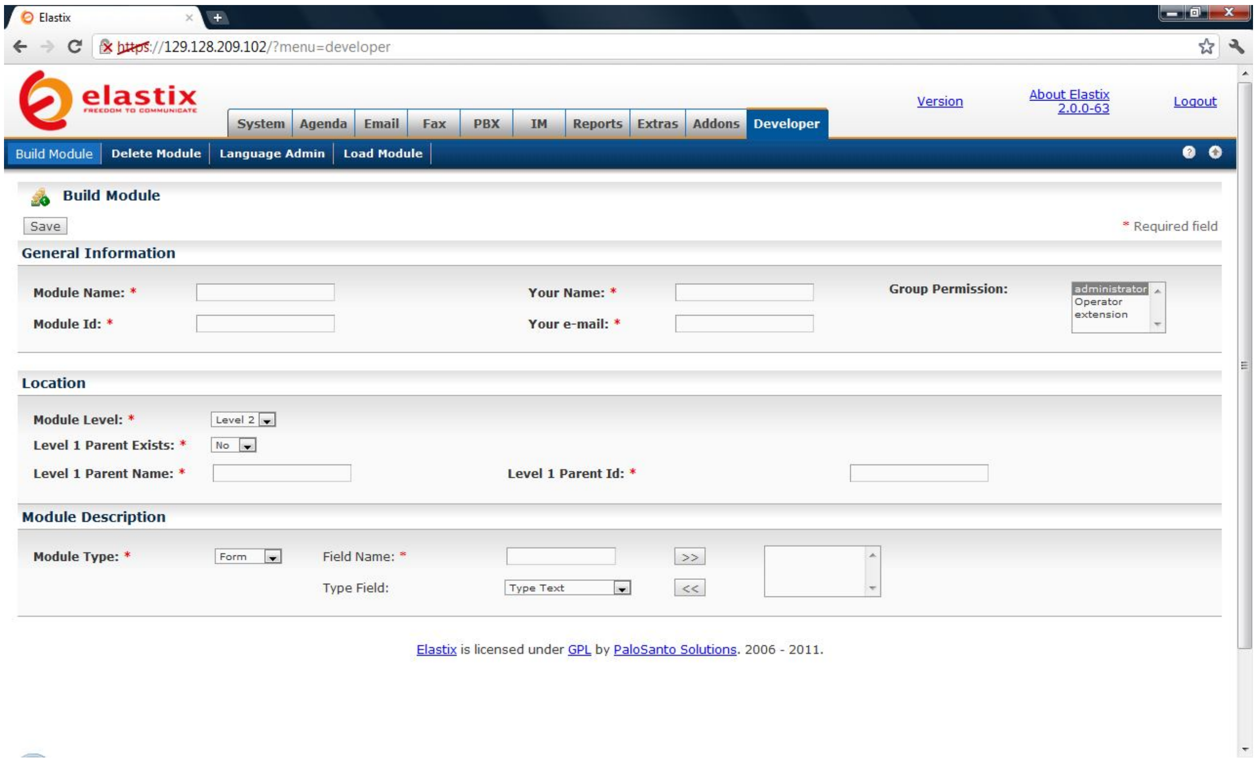
3.2.2 Installation of Developer module in Elastix :

The Elastix GUI development depends on the installation of Developer module in Elastix. The developer module is installed in the following two methods:

1. Via console in the Elastix CentOS server by issuing the following command :
yum install elastix-developer

2. Through Elastix web interface:
Navigate to the Addons menu to search for the Developer module, and select on Install as seen in the addons list .If the developer module is not present in the install list, it needs to be downloaded from the Elastix web site or the first method needs to be followed.

The following diagram shows how the developer module appears when it is installed in the Elastix GUI:



Build Module

* Required field

General Information

Module Name: *		Your Name: *		Group Permission:	administrator, Operator, extension
Module Id: *		Your e-mail: *			

Location

Module Level: *	Level 2		
Level 1 Parent Exists: *	No		
Level 1 Parent Name: *		Level 1 Parent Id: *	

Module Description

Module Type: *	Form	Field Name: *	
		Type Field:	Type Text

Elastix is licensed under GPL by PaloSanto Solutions. 2006 - 2011.

Fig 3.2.2.2: Developer module in Elastix

There are different classifications of the module such as Build Module, Delete module and Load module etc

The build module is further divided into three parts namely General Information, Location and Module Description. The General information section mainly consists of mandatory fields with different access level privileges:

Module name: The name of the module which need not be unique but must be a name which can be easily identified by the developers/users
Module Id: The name of the module folder, which must always be unique.
Name: Name of the person who is creating the new module.
E-mail: Email address of the developer who is responsible for the module
Group Permission: Access level privileges of the module whether it is accessible by administrator or operator etc

Module Level: It can select any of the three available levels namely Level-1, Level-2 and Level-3 etc

When Level-1 Parent is present, we define if the Level-1 parent menu exists in the newly to be created module. When we select the option “Yes”, a drop-down list with all available Level-1 menus will be displayed. When we select the option”No”, it will be displayed as only Level-1 menu. Similarly, if the module level selected as Level-2, then Level-1 parent name and Level-1 parent id will be displayed. If the module level selected is Level 3, then Level-1/Level-2 name and both parent id will be displayed.

Module Description:

The module types can be one from the selection menus like Form, Grid or Framed etc. The individual field name can be selected from the type field consisting of the drop-down list of text, date, password, radio, checkbox etc

Accordingly depending on the number of fields required for the module based on whether it is form, grid or framed can be selected based on the requirements of the developer when creating the new module.

By creating a new module through the Build Module page, the following tasks will be completed:

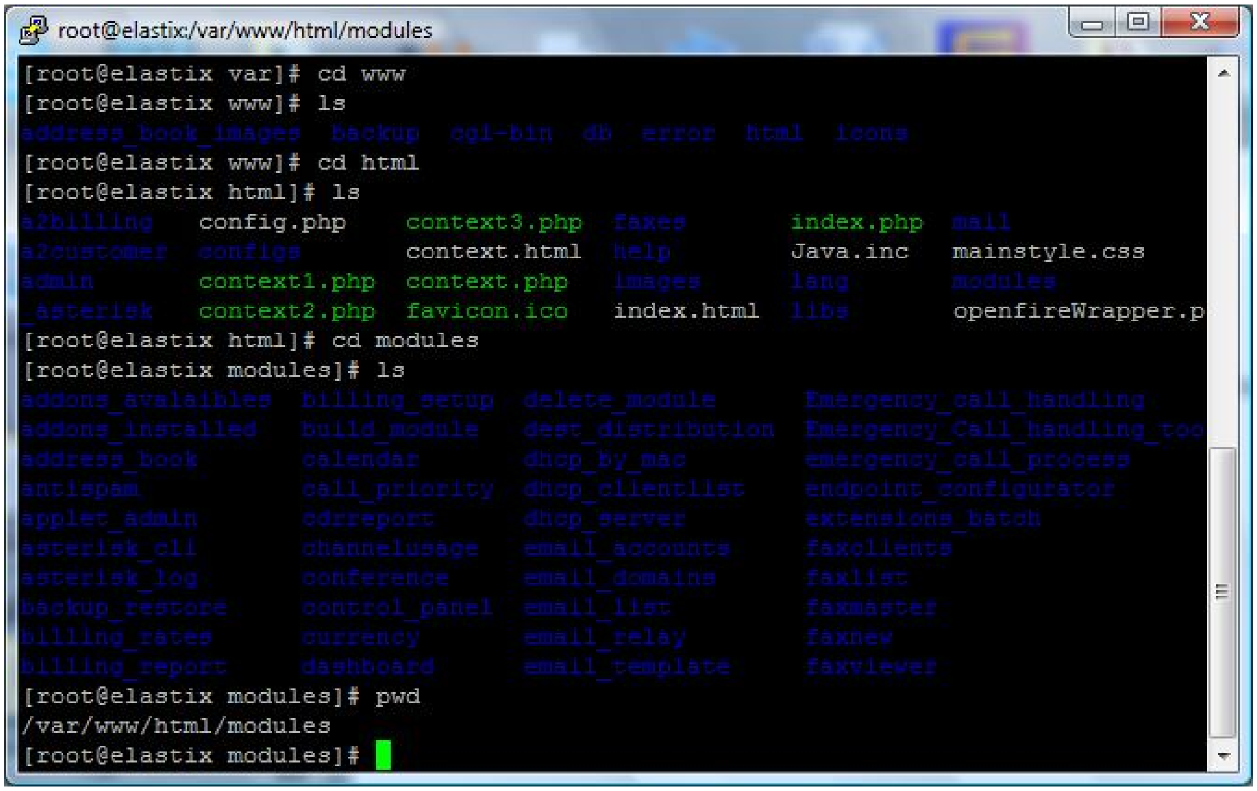
1. Creation of the module menu where the new module will be placed into. This can be at level-1 menu and the Level-2 menu depending on the requirements of the developer developing the module.

2. Design and development of the programming/folder hierarchy, located in the modules folder within the Elastix framework. The corresponding module folder will have as name the id given from the Build Module interface, which should be unique and should not be repeated.

A newly created module has the architecture of MVC2 module framework where:

a)configs: Contains the configuration file needed for new module that is being created . Whenever the database settings need to be implemented, the changes will be defined in this section
b)help: Contains all the user documentation/help required for the development and modification of the new and existing modules
c)images: Required for the appearance of the module in Elastix server
d)index.php: This is the main layer module where the script can be suitably modified. This is the main PHP script for the application which controls the behavior of the newly created elastix module.It is a script where further development can be carried out to embed additional features into the elastix web GUI module.
e)lang: Different language transitions based on regional language requirements of the module can be made in this folder.
f)libs: All libraries related to the module will be defined here. It basically acts as the model layer of the MVC2 framework.
g)themes: The module layout and the associated functional appearances can be defined in this module. It acts as the view layer of the MVC2 framework.

The modules will be stored in /var/www/modules folder of the Elastix web server. Further development of the module is restricted to making changes and doing new development within the newly created module folder of the Elastix server.



root@elastix:/var/www/html/modules

```
[root@elastix var]# cd www
[root@elastix www]# ls
address_book_images  backup  cgi-bin  db  error  html  icons
[root@elastix www]# cd html
[root@elastix html]# ls
a2billing   config.php    context3.php  faxes        index.php  mail
a2customer  configs       context.html  help         Java.inc   mainstyle.css
admin       context1.php  context.php   images       lang       modules
_asterisk   context2.php  favicon.ico   index.html   libs       openfireWrapper.p
[root@elastix html]# cd modules
[root@elastix modules]# ls
addons_availables  billing_setup  delete_module      Emergency_call_handling
addons_installed   build_module   dest_distribution  Emergency_Call_handling_too
address_book       calendar       dhcp_by_mac        emergency_call_process
antispam           call_priority  dhcp_clientlist    endpoint_configurator
applet_admin       cdrreport      dhcp_server        extensions_batch
asterisk_cli       channelusage   email_accounts     faxclients
asterisk_log       conference     email_domains      faxlist
backup_restore     control_panel  email_list         faxmaster
billing_rates      currency       email_relay        faxnew
billing_report     dashboard      email_template     faxviewer
[root@elastix modules]# pwd
/var/www/html/modules
[root@elastix modules]#
```

Fig 3.2.2.3: Presence of newly created modules inside the Elastix server

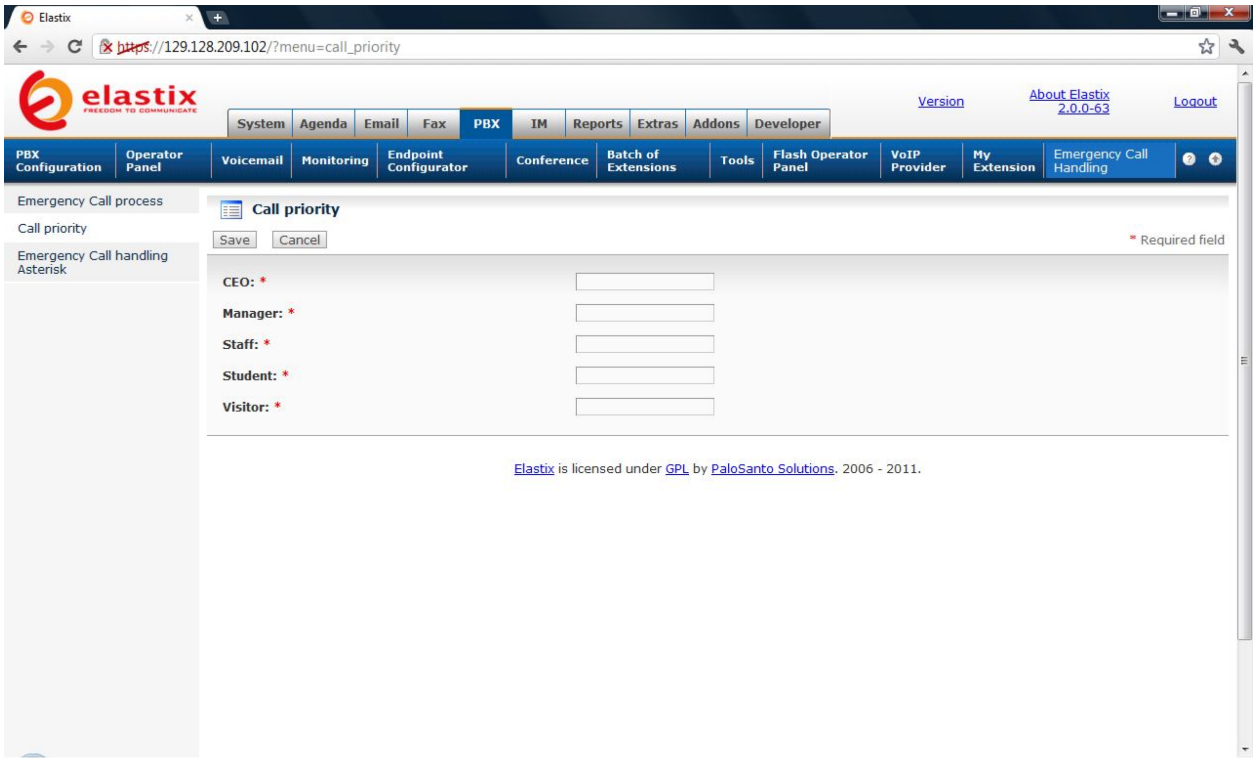
As shown in the above diagram, /var/www/html/modules contain all the modules that make up the Elastix web server.

3.2.3 Creation of Emergency call handling application:

The following are the steps followed in creating the emergency call handling application:

3.2.3.1 Creation of Emergency priority group:

One of the primary benefits of using the Elastix GUI is to have a greater control over managing the priorities and the contexts of user groups like CEO, Student, Visitor, Manager etc.



Call priority

* Required field

CEO: *	
Manager: *	
Staff: *	
Student: *	
Visitor: *	

Elastix is licensed under GPL by PaloSanto Solutions. 2006 - 2011.

Fig 3.2.3.1: Call priority module in Elastix

The Elastix GUI offers many features with creation of new PBX extensions, assigning the context priorities with facilities to use feature codes, inbound and outbound routes/call control with additional options like conference settings, voicemail and system recordings, IVR ring groups etc.

The Call priority module in Elastix was designed to enable additional features to the users to set priorities from the Elastix GUI. The Algorithm developed by the previous developer had these context priorities defined in the form of cost channel variable in the macro-dial trunkouts found in the dial plans of the extension_additional.conf files in the Elastix server:

```
[macro-dialout-trunk]

exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"],Set,COST=600)
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Manager"],Set,COST=500)
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Staff"],Set,COST=400)
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Student"],Set,COST=200)
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Visitor"],Set,COST=100)
```

This resulted in the need for the creation of the separate module for call priority so that the novice user from the Elastix GUI would be able to set priorities.i.e cost variables whose values would be accepted and set for global channel cost variable defined in the macro-dialout trunk.

Significant analysis was done on the feasibility of setting the priority in the form of Cost variables from Elastix GUI. As a result a module was developed that is shown in the above snapshot, which will accept the cost values for CEO, Manager, Staff, Student, Visitor etc. The call priority module is in the PHP and hence the PHP code had to establish a link with Asterisk AGI scripting since the algorithm that was developed by the previous developer only works on the basis of assigning values to COST variables in the macro-dial out trunk.

It was found that on Elastix offers very little flexibility when it comes to dealing with local variables (AGI variables) and Global variables (specific to Macros-in Dial plans).To gather more information and perform more analysis on these drawbacks as a result, a separate PHP script was designed. A external HTML form was created and PHP script was made to run once the data is submitted to the HTML form. This PHP script had the AGI script being embedded inside it and was mainly used for checking the assignment and comparison of cost variables. It was found that in PHP-AGI Scripting, the only way to assign values to the variables is by using the command:

```
AGI->set_variable(“variable’’,’’value’’);
```

So a local variable namely $cost was created inside the PHP script and value was assigned to it based on what users enter into the HTML form from the Elastix GUI. Accordingly,

`AGI->set_variable(“Cost”,”${cost}”)` where $cost is the local variable defined in the PHP script.

In the Macro-dialout-trunk ,it was defined as follows:

```
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"],Set,COST=${Cost})
```

In this way efforts were made to assign the value generated from the local variables into the Global variables defined in the Macro-dialout trunk. But it was found that the command :

`AGI->set_variable(“Cost”,”value”)` accepts only the actual values manually entered in the script and not with the values assigned to the local variables.

This is because AGI Scripting requires its own separate set of commands to operate. Although it can be embedded inside the PHP script to make it operate, it has its own syntax restrictions and scripting language limitations.

Hence, the following steps were taken to find a suitable solution:

The context.php script was run as an AGI script in the macro-dialout trunk and the $Cost value from the PHP script was passed onto the Global variable ‘COST’ in the macro-dialout trunk.

```
[macro-dialout-trunk]
include => macro-dialout-trunk-custom
exten => s,1,Set(DIAL_TRUNK=${ARG1})
exten => s,n(agi),Agi(/var/www/html/context.php)
exten => s,n,NoOp(Returned new priority:${PRIORITY})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"] ,Set,COST=${Cost})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Manager"],Set,COST=$[${Cost}-100])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Staff"],Set,COST=$[${Cost}-200])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Student"],Set,COST=$[${Cost}-300])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Visitor"],Set,COST=$[${Cost}-400])
```

One of the user groups have been assigned a fixed value in the AGI Script and based on the value that is being passed out to the macro-dialout-trunk,the cost factor of all other groups will get assigned before Emergency application runs when users dial 911 in a busy congested PBX environment with all other non-911 calls being in progress.

The CEO is having the highest cost priority and based on the normal cost priority hierarchy, the cost priorities of all other user groups are defined in the macro-dialout trunk.

The following results are seen in the Asterisk GUI :

```
Executing [21930@CEO:5] Macro("SIP/300-00000046", "dialout-trunk,1,21930,,") in new stack
    -- Executing [s@macro-dialout-trunk:1] Set("SIP/300-00000046", "DIAL_TRUNK=1") in new
stack
    -- Executing [s@macro-dialout-trunk:2] AGI("SIP/300-00000046",
"/var/www/html/context.php") in new stack
    -- Launched AGI Script /var/www/html/context.php
    -- <SIP/300-00000046>AGI Script /var/www/html/context.php completed, returning 0
    -- Executing [s@macro-dialout-trunk:3] NoOp("SIP/300-00000046", "Returned new priority:3")
in new stack
    -- Executing [s@macro-dialout-trunk:4] ExecIf("SIP/300-00000046", "1,Set,COST=700") in
new stack
    -- Executing [s@macro-dialout-trunk:5] ExecIf("SIP/300-00000046", "0,Set,COST=600") in
new stack
    -- Executing [s@macro-dialout-trunk:6] ExecIf("SIP/300-00000046", "0,Set,COST=500") in
new stack
    -- Executing [s@macro-dialout-trunk:7] ExecIf("SIP/300-00000046", "0,Set,COST=400") in
new stack
    -- Executing [s@macro-dialout-trunk:8] ExecIf("SIP/300-00000046", "0,Set,COST=300") in
new stack
    -- Executing [s@macro-dialout-trunk:9] GosubIf("SIP/300-00000046", "0?sub-pincheck,s,1")
in new stack
    -- Executing [s@macro-dialout-trunk:10] GotoIf("SIP/300-00000046", "0?disabletrunk,1") in
new stack
    -- Executing [s@macro-dialout-trunk:11] Set("SIP/300-00000046", "DIAL_NUMBER=21930")
in new stack
    -- Executing [s@macro-dialout-trunk:12] Set("SIP/300-00000046",
"DIAL_TRUNK_OPTIONS=tr") in new stack
    -- Executing [s@macro-dialout-trunk:13] Set("SIP/300-00000046",
"OUTBOUND_GROUP=OUT_1") in new stack
    -- Executing [s@macro-dialout-trunk:14] GotoIf("SIP/300-00000046", "1?nomax") in new
stack
```

In this way, the different user-groups will get assigned their cost priority values in the macro-dialout-trunk before the Emergency call handling algorithm comes into force in the Emergency call handling scenario.

This impending issue needs to be addressed with sufficient PHP and Asterisk(AGI) expertise and is considered in the future scope of work in the Emergency call handling application.

The mapping of assigning the cost values to newly created extensions and updating it in the call priority module is recommended as future work and it is described in detail in the Future scope section of the document.

3.2.3.2 Designing the AGI script (Making the AGI Scripts to consistently run in the background):

The main algorithm is developed in the Java programming language. The code resides in the /etc/asterisk folder of the Elastix server.The AGI Script invokes the Java program when there is Emergency.ie [Emergency] defined in the extensions.conf file and the Emergency Dial plan gets into act .In that scenario, proper matching of AGI script and the java program is achieved with the help of fastagi mapping properties file. For the AGI Script to consistently run, DefaultAgiServer needs to be started and must be continuously running as a cron job in the background. It is achieved using the following command :

```
[root@elastix asterisk]## java -cp asterisk-java-1.0.0.M3.jar:.
org.asteriskjava.fastagi.DefaultAGIServer

Dec16,2011 5:33:27 PM org.asteriskjava.fastagi.DefaultAgiServer startup

INFO:Listening on *:4573.
```

This command should be made to run in the background after the Emergency call module is installed in the Elastix server. This triggers the defaultAGIServer to get started and maps the classpath of Asterisk-Java jar file. When the Emergency situation occurs and user groups dial in, then the AGI server which is running in the background accepts the incoming connections and emergency dialplan takes care of the AGI script that executes by calling the java program (algorithm) and the AGI script hangs up the call in the channel having the lowest cost factor.

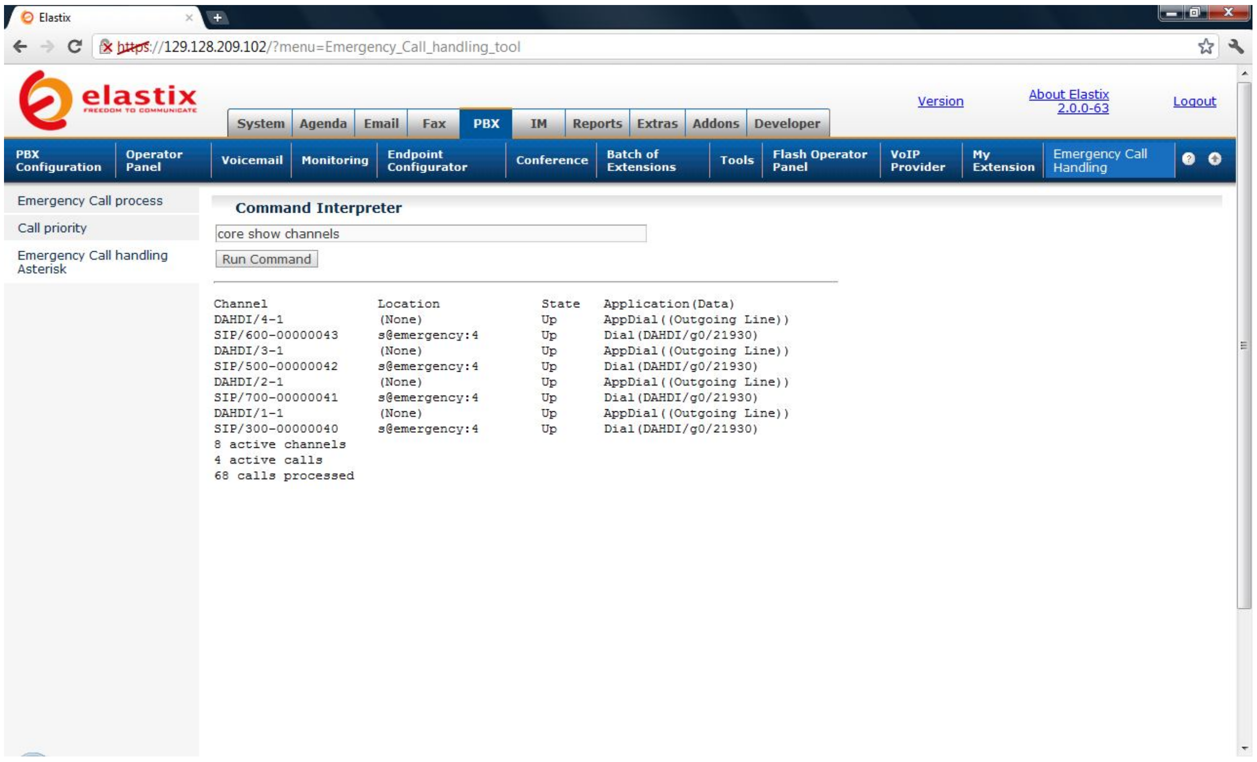
```
# java -cp asterisk-java-1.0.0.M3.jar:. org.asteriskjava.fastagi.DefaultAGIServer &

[root@elastix asterisk]# java -cp asterisk-java-1.0.0.M3.jar:.
org.asteriskjava.fastagi.DefaultAgiServer

17-Dec-2011 5:19:12 PM org.asteriskjava.fastagi.DefaultAgiServer startup

INFO: Listening on *:4573.

17-Dec-2011 5:20:28 PM org.asteriskjava.fastagi.DefaultAgiServer startup

INFO: Received connection from /127.0.0.1

17-Dec-2011 5:20:28 PM org.asteriskjava.fastagi.AbstractAgiServer getPool

INFO: Thread pool started.
```

The DefaultAGIServer is made to run in the background as a cron job and depending on user requirements and suitability they can be restarted on a daily, weekly or monthly basis. This will make sure the Emergency call handling application is continuously running in the background.

3.2.3.3 Creation of Emergency Call Handling Asterisk:

The Emergency call handling application consists of Emergency call handling shell in which the novice user from the Elastix GUI can enter and run the Asterisk commands from the GUI itself. The user can enter all valid commands related to the Asterisk operation. The valid Asterisk command list is found in the help section of the Emergency Call Handling Asterisk module of the application.

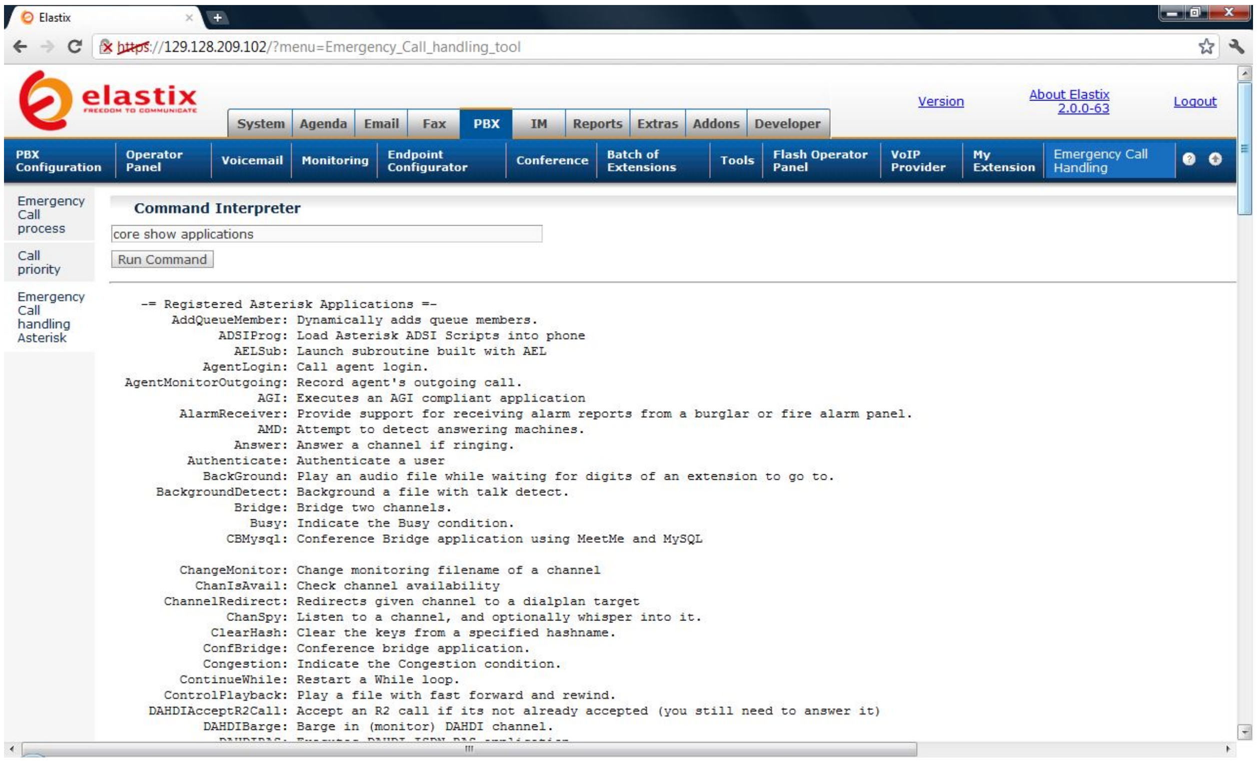
Below is the snapshot of the Command interpreter when the application is running and there are 4 user groups dialing the emergency number 911 :



```
core show channels
```

```
Channel             Location            State    Application(Data)
DAHDI/4-1           (None)              Up       AppDial((Outgoing Line))
SIP/600-00000043    s@emergency:4       Up       Dial(DAHDI/g0/21930)
DAHDI/3-1           (None)              Up       AppDial((Outgoing Line))
SIP/500-00000042    s@emergency:4       Up       Dial(DAHDI/g0/21930)
DAHDI/2-1           (None)              Up       AppDial((Outgoing Line))
SIP/700-00000041    s@emergency:4       Up       Dial(DAHDI/g0/21930)
DAHDI/1-1           (None)              Up       AppDial((Outgoing Line))
SIP/300-00000040    s@emergency:4       Up       Dial(DAHDI/g0/21930)
8 active channels
4 active calls
68 calls processed
```

Fig 3.2.3.3 : Elastix Asterisk Emergency call handling command Interpreter module



core show applications

```
   -= Registered Asterisk Applications =-
       AddQueueMember: Dynamically adds queue members.
             ADSIProg: Load Asterisk ADSI Scripts into phone
               AELSub: Launch subroutine built with AEL
           AgentLogin: Call agent login.
 AgentMonitorOutgoing: Record agent's outgoing call.
                  AGI: Executes an AGI compliant application
        AlarmReceiver: Provide support for receiving alarm reports from a burglar or fire alarm panel.
                  AMD: Attempt to detect answering machines.
               Answer: Answer a channel if ringing.
         Authenticate: Authenticate a user
           BackGround: Play an audio file while waiting for digits of an extension to go to.
      BackgroundDetect: Background a file with talk detect.
               Bridge: Bridge two channels.
                 Busy: Indicate the Busy condition.
              CBMysql: Conference Bridge application using MeetMe and MySQL

        ChangeMonitor: Change monitoring filename of a channel
          ChanIsAvail: Check channel availability
      ChannelRedirect: Redirects given channel to a dialplan target
              ChanSpy: Listen to a channel, and optionally whisper into it.
            ClearHash: Clear the keys from a specified hashname.
           ConfBridge: Conference bridge application.
           Congestion: Indicate the Congestion condition.
        ContinueWhile: Restart a While loop.
      ControlPlayback: Play a file with fast forward and rewind.
    DAHDIAcceptR2Call: Accept an R2 call if its not already accepted (you still need to answer it)
           DAHDIBarge: Barge in (monitor) DAHDI channel.
```

Fig 3.2.3.4: Demonstration of existing registered applications in Asterisk

The Elastix call handling command interpreter offers unmatched ability for a user to perform any of the tasks related to the application. Below are some of the simple tasks that the user may be able to perform when he encounters some issues while working with the application :

a) Restart the Asterisk server when required
b) Perform any database operations of the Asterisk database
c) Operate Dialplans from the GUI and can add and remove the extensions with the help of the dialplan commands
d) Enable Gtalk/Skype or Jabber debugging when there are issues of integration of instant messengers with Elastix
e) Perform DUNDi debugging and get a complete overview on the functionality of the Asterisk through Elastix GUI.

The important benefits of using the Command Interpreter for the Emergency call handling application is given below:

1) When the emergency call handling is in process, it can be used to check whether the application is running successfully and whether all the lines when dialing 911 are busy.

2) It can be used to check the important processes running in the application and also whether the application is running successfully in the background by checking PHP and java related scripts and other processes.

3) It is used to run AGI commands in the Asterisk thereby checking whether the AGI Scripts are also running successfully or not.

4) An important usage is the dialplan reload command that can be used in the command interpreter. Most of the times , changes related to Emergency call handling application are done in the dialplan in extensions.conf,globals_custom.conf and extensions_additional.conf files.Dialplan reload command helps to reload the dialplan whenever the changes are made to the application. This saves time and prevents the user to contact administrators to run those commands and to restart the asterisk server everytime when they encounter an issue with the application.

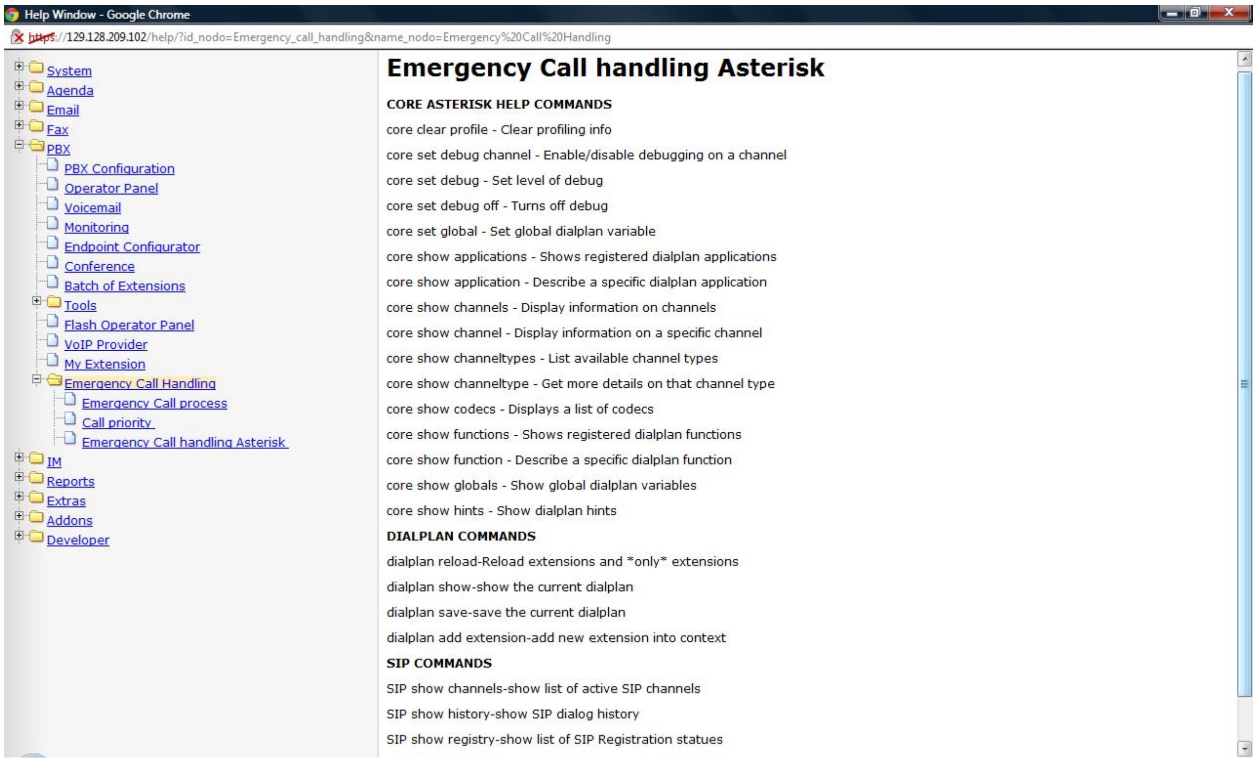
5) Manually perform the operation of requesting a hang up when desired by the user for any debugging purposes.

The brief description of how these tasks are accomplished is explained in the testing scenarios section of the current document:

3.2.4 Help Section of the modules in the Elastix GUI :

One of the important features of the Elastix GUI is the user support in the form of documentation. Each module consists of help section which describe all the information necessary to operate the individual module or the entire emergency call handling application.

When we consider the Emergency Call handling asterisk module, the help section provides user the important information on the list of commands that can be used to check the application and even the functionality of the Asterisk server.



Emergency Call handling Asterisk

CORE ASTERISK HELP COMMANDS

core clear profile - Clear profiling info

core set debug channel - Enable/disable debugging on a channel

core set debug - Set level of debug

core set debug off - Turns off debug

core set global - Set global dialplan variable

core show applications - Shows registered dialplan applications

core show application - Describe a specific dialplan application

core show channels - Display information on channels

core show channel - Display information on a specific channel

core show channeltypes - List available channel types

core show channeltype - Get more details on that channel type

core show codecs - Displays a list of codecs

core show functions - Shows registered dialplan functions

core show function - Describe a specific dialplan function

core show globals - Show global dialplan variables

core show hints - Show dialplan hints

DIALPLAN COMMANDS

dialplan reload-Reload extensions and *only* extensions

dialplan show-show the current dialplan

dialplan save-save the current dialplan

dialplan add extension-add new extension into context

SIP COMMANDS

SIP show channels-show list of active SIP channels

SIP show history-show SIP dialog history

SIP show registry-show list of SIP Registration statues

Fig 3.2.4 :Emergency call handling Asterisk Commands in the help section of the module

The user documentation is prepared in the /var/www/html/modules/Emergency_call_handling_tool/help folder of the Elastix server in the form of HTML (hlp) files.

4. Implementation Process and Testing:

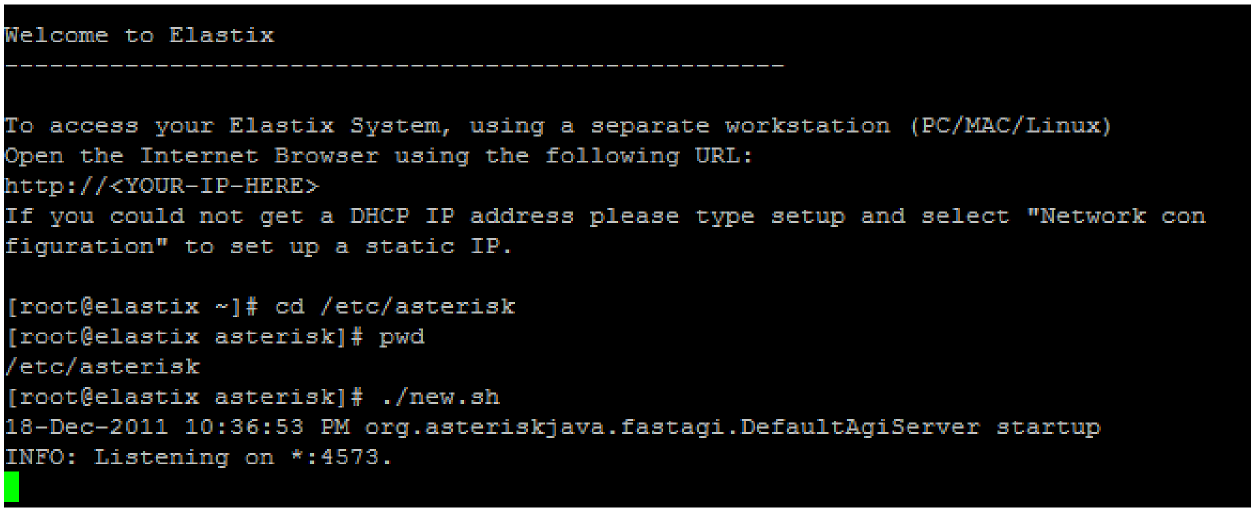
This section describes the implementation process and the sequence of Events that will happen when the Emergency call process is in place.

The Emergency call handling application script runs only when the Emergency call process situation occurs.ie. when there is the need to release a channel with low cost priority. In order for the script to run in the background, the DefaultAGI server must always be started and sequence of Asterisk events when the Asterisk calls are made must be running in sync with existing Java and PHP /Asterisk set up. The following section demonstrates how the script continuously runs in the background:

In the /etc/asterisk/ folder of the Elastix server, the DefaultAGI server must be started using the following command :

# java -cp asterisk-java-1.0.0.M3.jar:. org.asteriskjava.fastagi.DefaultAGIServer

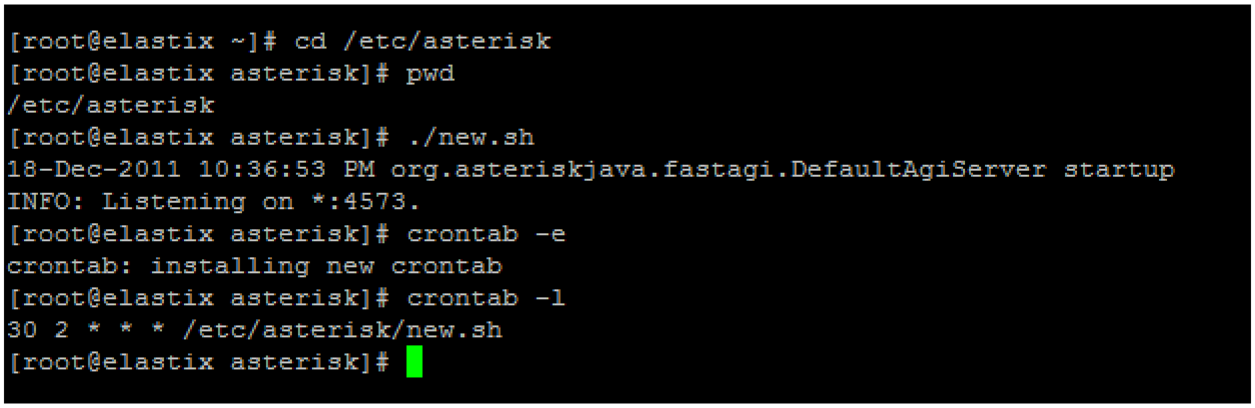
A Shell script *new.sh* is developed and resides inside the /etc/asterisk folder and it contains the above command and whenever the shell script gets executed the DefaultAGIServer gets started.



```
Welcome to Elastix
----------------------------------------------------

To access your Elastix System, using a separate workstation (PC/MAC/Linux)
Open the Internet Browser using the following URL:
http://<YOUR-IP-HERE>
If you could not get a DHCP IP address please type setup and select "Network con
figuration" to set up a static IP.

[root@elastix ~]# cd /etc/asterisk
[root@elastix asterisk]# pwd
/etc/asterisk
[root@elastix asterisk]# ./new.sh
18-Dec-2011 10:36:53 PM org.asteriskjava.fastagi.DefaultAgiServer startup
INFO: Listening on *:4573.
```

Fig 4.1:Shell script starting the DefaultAgiServer

A crontab entry is made in the CentOS server of the Elastix to run this process once in a day.It can be seen using the command :



```
[root@elastix ~]# cd /etc/asterisk
[root@elastix asterisk]# pwd
/etc/asterisk
[root@elastix asterisk]# ./new.sh
18-Dec-2011 10:36:53 PM org.asteriskjava.fastagi.DefaultAgiServer startup
INFO: Listening on *:4573.
[root@elastix asterisk]# crontab -e
crontab: installing new crontab
[root@elastix asterisk]# crontab -l
30 2 * * * /etc/asterisk/new.sh
[root@elastix asterisk]#
```

Fig 4.1.1:Crontab entry seen in the CentOS server

The above snapshot shows an entry of crontab that runs daily once at 2-30 am.This will start the DefaultAGIServer and users can implement the emergency call handling process at any point of time in the Elastix GUI.This crontab entry can be suitably modified to run multiple number of times depending on the requirement and usage of the Emergency call handling application.

So ,when the users use the Elastix GUI and encounter an Emergency call process situation the main application script checks to find whether the DefaultAGI server is started and whether it is listening/receiving the incoming connections form the Asterisk Dialplan.The AGI application script then successfully runs calling the main application script which is in java and handles the Emergency call process by allowing a user group to dial 911 by releasing the call of a user group with lowest cost priority. This is explained below in the testing section of the Emergency call handling application.

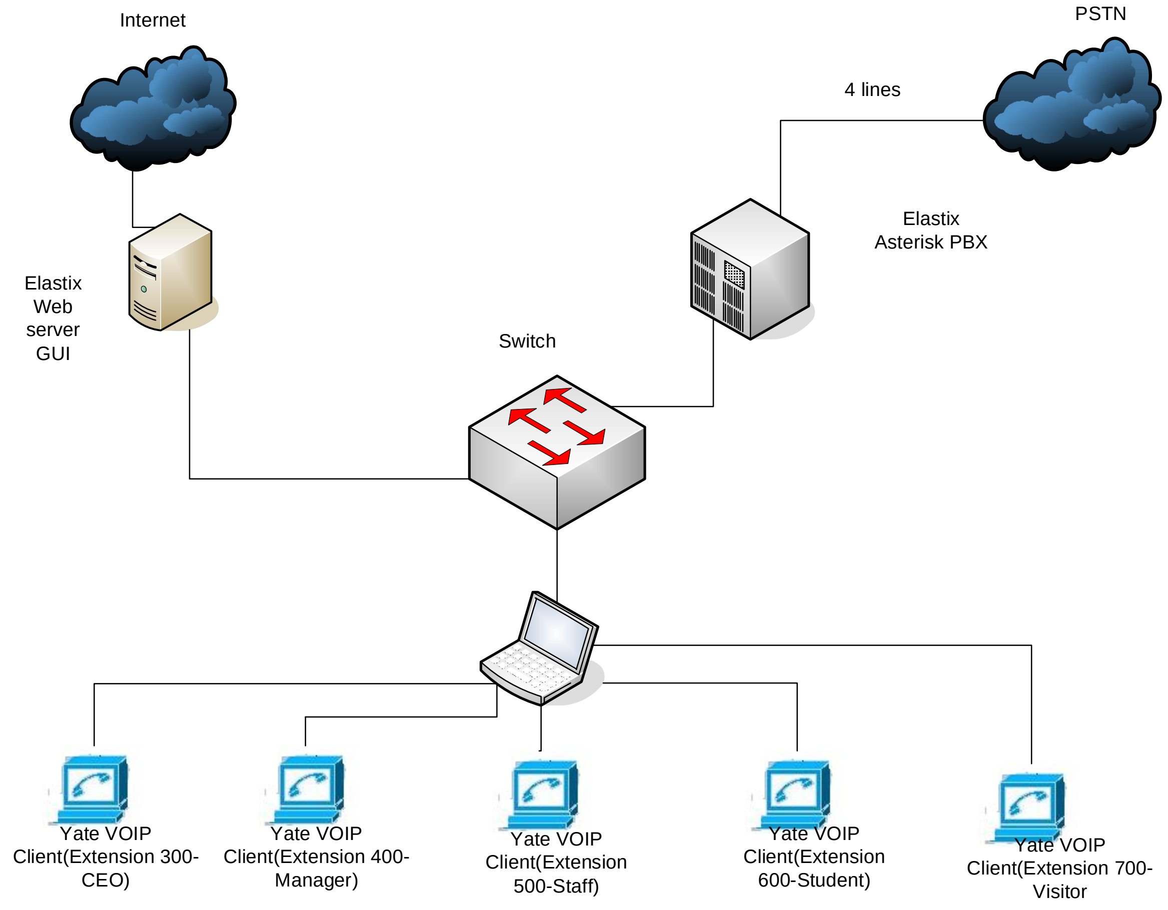
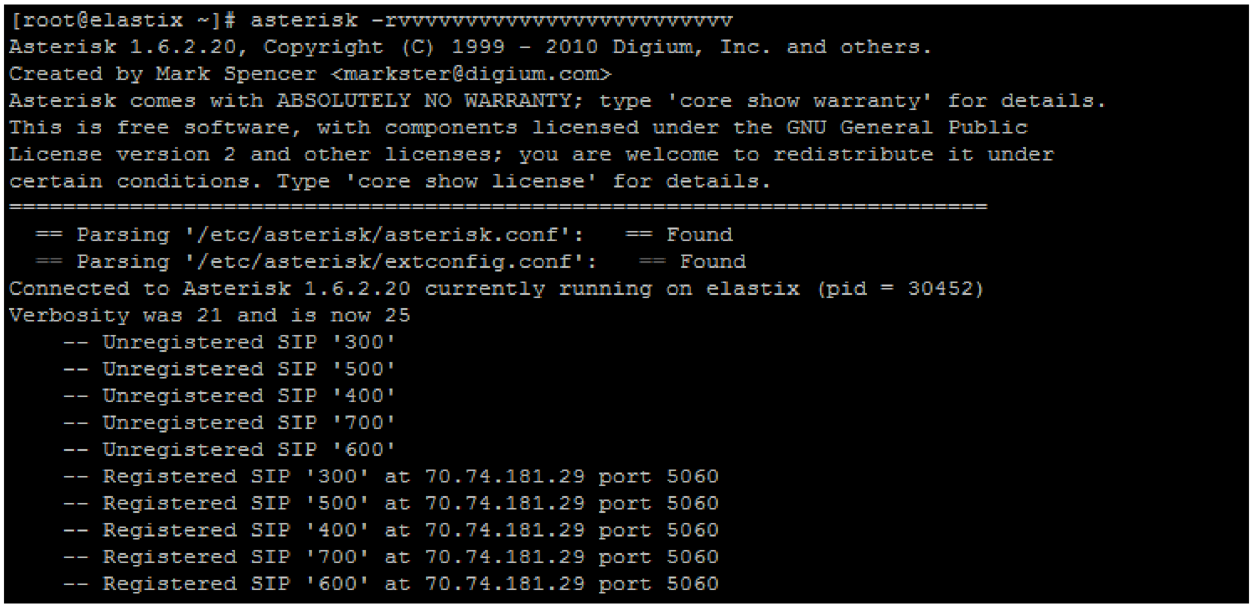

Internet
PSTN
4 lines
Elastix Asterisk PBX
Elastix Web server GUI
Switch
Yate VOIP Client(Extension 300-CEO)
Yate VOIP Client(Extension 400-Manager)
Yate VOIP Client(Extension 500-Staff)
Yate VOIP Client(Extension 600-Student)
Yate VOIP Client(Extension 700-Visitor

Fig 4.1.2 : Testing setup for running the Emergency Call Handling application

4.1 Implementation of Testing Environment and Setup:

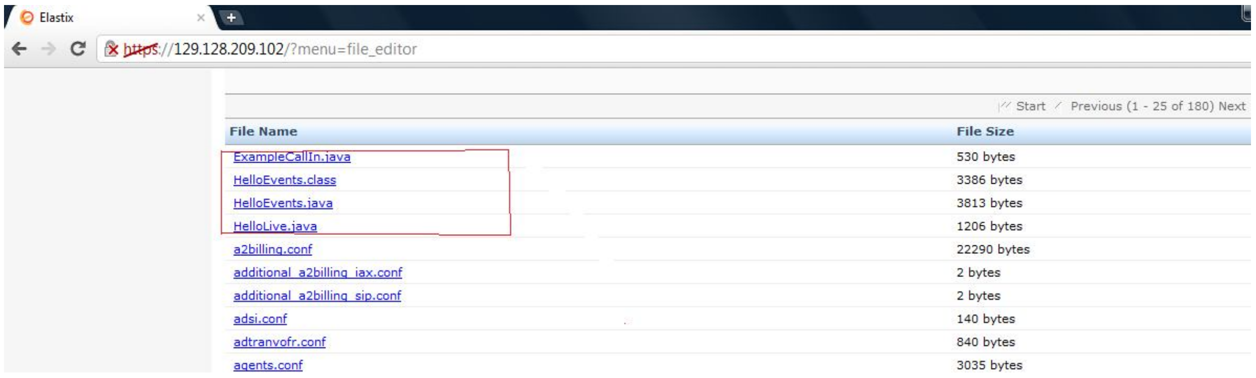
Five SIP accounts were registered in the form of YATE VoIP –soft phones and it was also registered in the Asterisk Server:



```
[root@elastix ~]# asterisk -rvvvvvvvvvvvvvvvvvvvvvvvvvvv
Asterisk 1.6.2.20, Copyright (C) 1999 - 2010 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
  == Parsing '/etc/asterisk/asterisk.conf':   == Found
  == Parsing '/etc/asterisk/extconfig.conf':   == Found
Connected to Asterisk 1.6.2.20 currently running on elastix (pid = 30452)
Verbosity was 21 and is now 25
    -- Unregistered SIP '300'
    -- Unregistered SIP '500'
    -- Unregistered SIP '400'
    -- Unregistered SIP '700'
    -- Unregistered SIP '600'
    -- Registered SIP '300' at 70.74.181.29 port 5060
    -- Registered SIP '500' at 70.74.181.29 port 5060
    -- Registered SIP '400' at 70.74.181.29 port 5060
    -- Registered SIP '700' at 70.74.181.29 port 5060
    -- Registered SIP '600' at 70.74.181.29 port 5060
```

Fig 4.1.3: Registered SIP clients (usergroups) with Extension numbers

The following important files were loaded in the /etc/asterisk folder of elastix server:



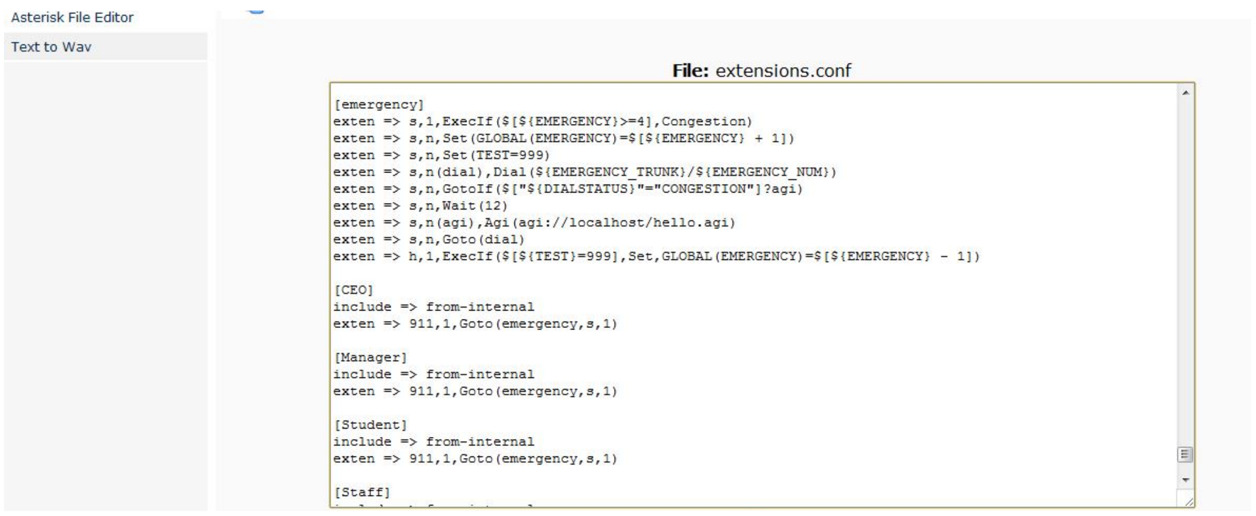
Elastix
~~https~~://129.128.209.102/?menu=file_editor
Start Previous (1 - 25 of 180) Next

File Name	File Size
ExampleCallIn.java	530 bytes
HelloEvents.class	3386 bytes
HelloEvents.java	3813 bytes
HelloLive.java	1206 bytes
a2billing.conf	22290 bytes
additional_a2billing_iax.conf	2 bytes
additional_a2billing_sip.conf	2 bytes
adsi.conf	140 bytes
adtranvofr.conf	840 bytes
agents.conf	3035 bytes

Fig 4.1.4 : Important files present inside the Elastix Asterisk server

The file HelloEvents.java is the main algorithm that is present inside the /etc/asterisk folder of the elastix server and it gets called by the AGI Script ***Hello.agi***.

In the Emergency Call handling situation it gets the call from the Asterisk in the form of the Argument agi://localhost/hello.agi that is defined in the dialplan [emergency] in the extensions.conf file.

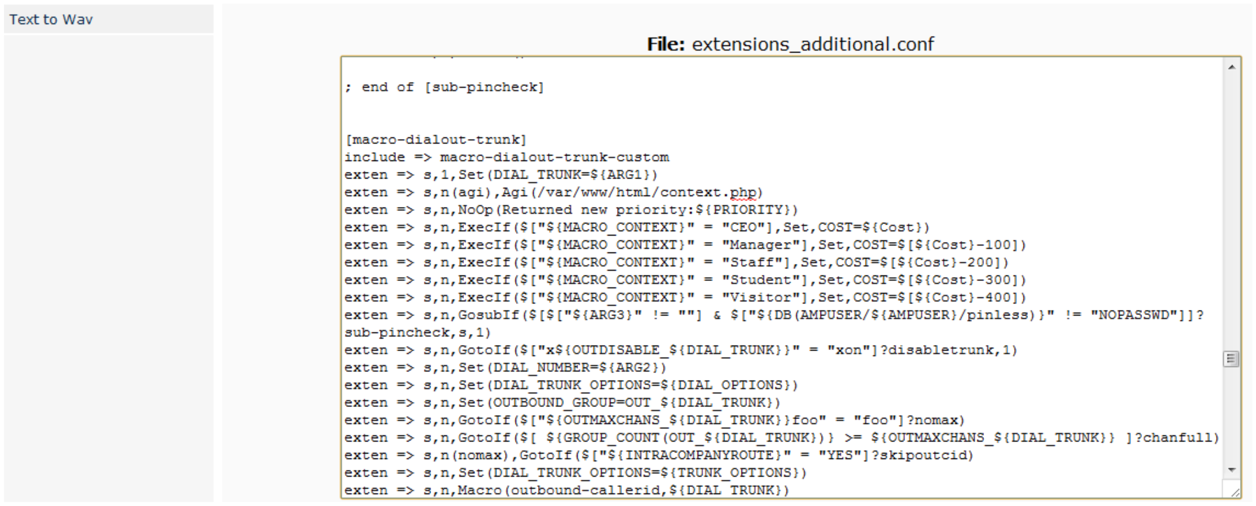


File: extensions.conf

```
[emergency]
exten => s,1,ExecIf($[${EMERGENCY}>=4],Congestion)
exten => s,n,Set(GLOBAL(EMERGENCY)=$[${EMERGENCY} + 1])
exten => s,n,Set(TEST=999)
exten => s,n(dial),Dial(${EMERGENCY_TRUNK}/${EMERGENCY_NUM})
exten => s,n,GotoIf($["${DIALSTATUS}"="CONGESTION"]?agi)
exten => s,n,Wait(12)
exten => s,n(agi),Agi(agi://localhost/hello.agi)
exten => s,n,Goto(dial)
exten => h,1,ExecIf($[${TEST}=999],Set,GLOBAL(EMERGENCY)=$[${EMERGENCY} - 1])

[CEO]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Manager]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Student]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Staff]
```

Fig 4.1.5 : Extensions.conf file in the Elastix Asterisk server

The macro dialout trunk is defined in the extensions_additional.conf file present in the Asterisk server:



File: extensions_additional.conf

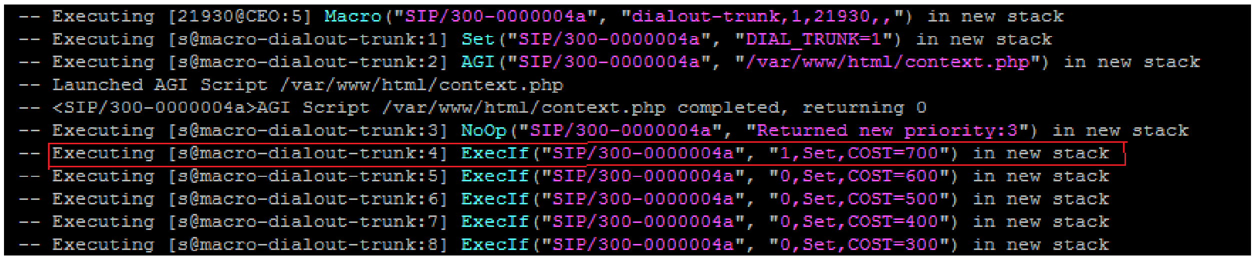
```
; end of [sub-pincheck]


[macro-dialout-trunk]
include => macro-dialout-trunk-custom
exten => s,1,Set(DIAL_TRUNK=${ARG1})
exten => s,n(agi),Agi(/var/www/html/context.php)
exten => s,n,NoOp(Returned new priority:${PRIORITY})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"],Set,COST=${Cost})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Manager"],Set,COST=$[${Cost}-100])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Staff"],Set,COST=$[${Cost}-200])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Student"],Set,COST=$[${Cost}-300])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Visitor"],Set,COST=$[${Cost}-400])
exten => s,n,GosubIf($[$["${ARG3}" != ""] & $["${DB(AMPUSER/${AMPUSER}/pinless)}" != "NOPASSWD"]]?
sub-pincheck,s,1)
exten => s,n,GotoIf($["x${OUTDISABLE_${DIAL_TRUNK}}" = "xon"]?disabletrunk,1)
exten => s,n,Set(DIAL_NUMBER=${ARG2})
exten => s,n,Set(DIAL_TRUNK_OPTIONS=${DIAL_OPTIONS})
exten => s,n,Set(OUTBOUND_GROUP=OUT_${DIAL_TRUNK})
exten => s,n,GotoIf($["${OUTMAXCHANS_${DIAL_TRUNK}}foo" = "foo"]?nomax)
exten => s,n,GotoIf($[ ${GROUP_COUNT(OUT_${DIAL_TRUNK})} >= ${OUTMAXCHANS_${DIAL_TRUNK}} ]?chanfull)
exten => s,n(nomax),GotoIf($["${INTRACOMPANYROUTE}" = "YES"]?skipoutcid)
exten => s,n,Set(DIAL_TRUNK_OPTIONS=${TRUNK_OPTIONS})
exten => s,n,Macro(outbound-callerid,${DIAL_TRUNK})
```

Fig 4.1.6:Extensions_additional.conf file in the Elastix Asterisk server

4.2 Execution of Test scenarios:

The Emergency call handling application is tested in the following manner :

The PHP script context.php has the fixed cost “700” being assigned to the CEO user group (Extension:300) when it gets executed successfully in the macro-dialout trunk. The cost assignment of other groups can be seen in the macro-dialout trunk:

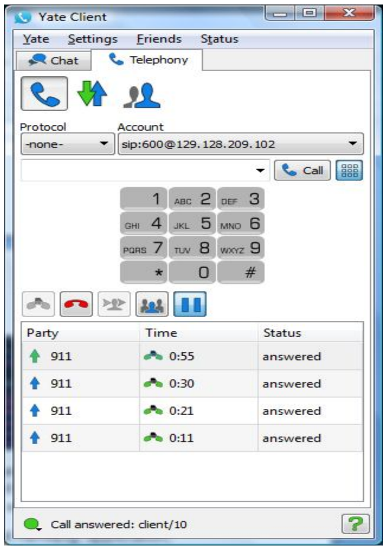


```
-- Executing [21930@CEO:5] Macro("SIP/300-0000004a", "dialout-trunk,1,21930,,") in new stack
-- Executing [s@macro-dialout-trunk:1] Set("SIP/300-0000004a", "DIAL_TRUNK=1") in new stack
-- Executing [s@macro-dialout-trunk:2] AGI("SIP/300-0000004a", "/var/www/html/context.php") in new stack
-- Launched AGI Script /var/www/html/context.php
-- <SIP/300-0000004a>AGI Script /var/www/html/context.php completed, returning 0
-- Executing [s@macro-dialout-trunk:3] NoOp("SIP/300-0000004a", "Returned new priority:3") in new stack
-- Executing [s@macro-dialout-trunk:4] ExecIf("SIP/300-0000004a", "1,Set,COST=700") in new stack
-- Executing [s@macro-dialout-trunk:5] ExecIf("SIP/300-0000004a", "0,Set,COST=600") in new stack
-- Executing [s@macro-dialout-trunk:6] ExecIf("SIP/300-0000004a", "0,Set,COST=500") in new stack
-- Executing [s@macro-dialout-trunk:7] ExecIf("SIP/300-0000004a", "0,Set,COST=400") in new stack
-- Executing [s@macro-dialout-trunk:8] ExecIf("SIP/300-0000004a", "0,Set,COST=300") in new stack
```

Fig 4.2.1 : Assignment of Cost for the CEO user group after execution of cost AGI inside context.php file

The above execution shows that a call from CEO user group was made and the cost value “700” was assigned from the PHP script context.php. Based on the cost value of CEO group and on the pattern in the macro-dialout trunk, the Cost values of all other groups were also assigned.

Now in order to showcase the working functionality of Emergency call handling application, Emergency 911 calls were made using Yate VoIP user client from the four user groups namely CEO, Manager, Staff, Student and the following snapshot shows that it accepted all four emergency calls and did not disconnect any of the emergency calls or did not respond to other 911 calls as busy when any one of the Emergency 911 call was in progress:

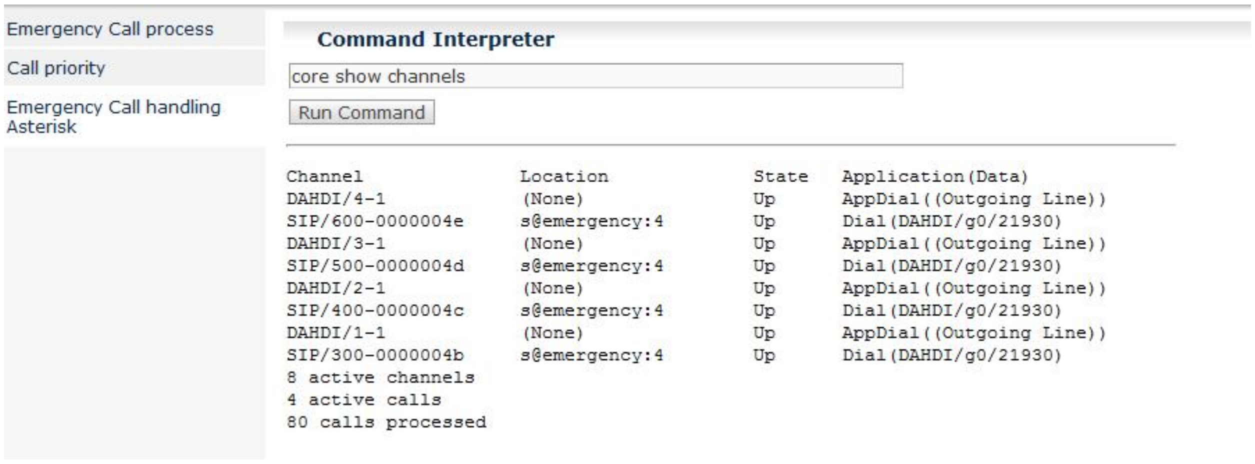


Party	Time	Status
911	0:55	answered
911	0:30	answered
911	0:21	answered
911	0:11	answered

Call answered: client/10

Fig 4.2.2: All lines dialing 911 and line circuits are busy with congestion

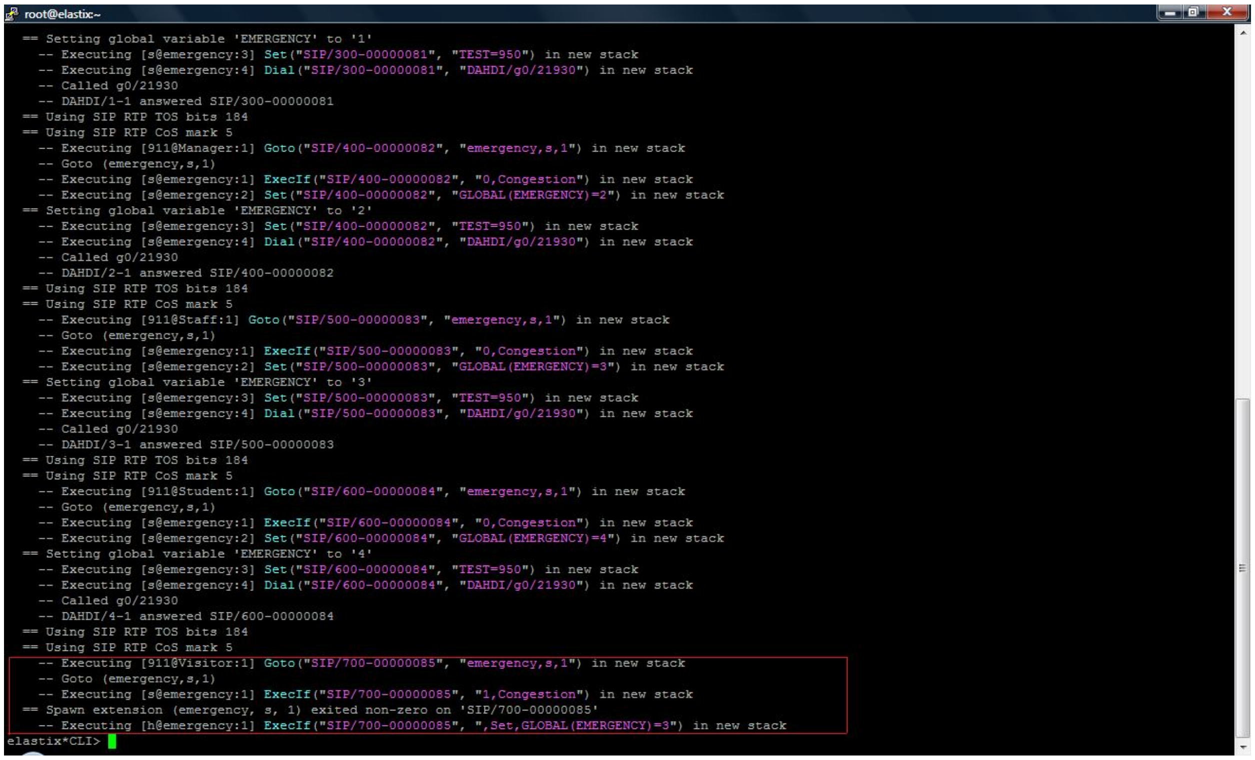
Now when the fifth 911 call was made an IVR response was initiated and only the cost assignment script got executed and no 911 call was disconnected from the elastix GUI. This indicates that the utmost preference is given to 911 emergency calls in the emergency call handling application and no 911 call will be disconnected under any circumstances. This was congestion as current setup supports only four concurrent 911 emergency calls at a time.



```
Channel              Location             State    Application(Data)
DAHDI/4-1            (None)               Up       AppDial((Outgoing Line))
SIP/600-0000004e     s@emergency:4        Up       Dial(DAHDI/g0/21930)
DAHDI/3-1            (None)               Up       AppDial((Outgoing Line))
SIP/500-0000004d     s@emergency:4        Up       Dial(DAHDI/g0/21930)
DAHDI/2-1            (None)               Up       AppDial((Outgoing Line))
SIP/400-0000004c     s@emergency:4        Up       Dial(DAHDI/g0/21930)
DAHDI/1-1            (None)               Up       AppDial((Outgoing Line))
SIP/300-0000004b     s@emergency:4        Up       Dial(DAHDI/g0/21930)
8 active channels
4 active calls
80 calls processed
```

Fig 4.2.3: Emergency call handling asterisk demonstrating the four emergency calls active in progress

This was confirmed in the Elastix server as shown in the above diagram.The user groups CEO(300),Manager(400),Staff(500),Student(600) were in progress and IVR response was initiated for the fifth 911 caller.



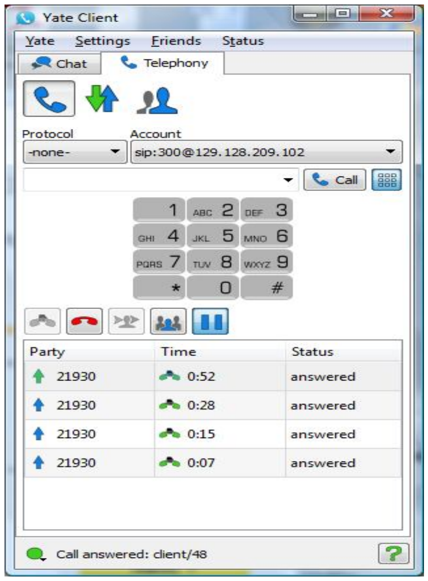
```
  == Setting global variable 'EMERGENCY' to '1'
    -- Executing [s@emergency:3] Set("SIP/300-00000081", "TEST=950") in new stack
    -- Executing [s@emergency:4] Dial("SIP/300-00000081", "DAHDI/g0/21930") in new stack
    -- Called g0/21930
    -- DAHDI/1-1 answered SIP/300-00000081
  == Using SIP RTP TOS bits 184
  == Using SIP RTP CoS mark 5
    -- Executing [911@Manager:1] Goto("SIP/400-00000082", "emergency,s,1") in new stack
    -- Goto (emergency,s,1)
    -- Executing [s@emergency:1] ExecIf("SIP/400-00000082", "0,Congestion") in new stack
    -- Executing [s@emergency:2] Set("SIP/400-00000082", "GLOBAL(EMERGENCY)=2") in new stack
  == Setting global variable 'EMERGENCY' to '2'
    -- Executing [s@emergency:3] Set("SIP/400-00000082", "TEST=950") in new stack
    -- Executing [s@emergency:4] Dial("SIP/400-00000082", "DAHDI/g0/21930") in new stack
    -- Called g0/21930
    -- DAHDI/2-1 answered SIP/400-00000082
  == Using SIP RTP TOS bits 184
  == Using SIP RTP CoS mark 5
    -- Executing [911@Staff:1] Goto("SIP/500-00000083", "emergency,s,1") in new stack
    -- Goto (emergency,s,1)
    -- Executing [s@emergency:1] ExecIf("SIP/500-00000083", "0,Congestion") in new stack
    -- Executing [s@emergency:2] Set("SIP/500-00000083", "GLOBAL(EMERGENCY)=3") in new stack
  == Setting global variable 'EMERGENCY' to '3'
    -- Executing [s@emergency:3] Set("SIP/500-00000083", "TEST=950") in new stack
    -- Executing [s@emergency:4] Dial("SIP/500-00000083", "DAHDI/g0/21930") in new stack
    -- Called g0/21930
    -- DAHDI/3-1 answered SIP/500-00000083
  == Using SIP RTP TOS bits 184
  == Using SIP RTP CoS mark 5
    -- Executing [911@Student:1] Goto("SIP/600-00000084", "emergency,s,1") in new stack
    -- Goto (emergency,s,1)
    -- Executing [s@emergency:1] ExecIf("SIP/600-00000084", "0,Congestion") in new stack
    -- Executing [s@emergency:2] Set("SIP/600-00000084", "GLOBAL(EMERGENCY)=4") in new stack
  == Setting global variable 'EMERGENCY' to '4'
    -- Executing [s@emergency:3] Set("SIP/600-00000084", "TEST=950") in new stack
    -- Executing [s@emergency:4] Dial("SIP/600-00000084", "DAHDI/g0/21930") in new stack
    -- Called g0/21930
    -- DAHDI/4-1 answered SIP/600-00000084
  == Using SIP RTP TOS bits 184
  == Using SIP RTP CoS mark 5
    -- Executing [911@Visitor:1] Goto("SIP/700-00000085", "emergency,s,1") in new stack
    -- Goto (emergency,s,1)
    -- Executing [s@emergency:1] ExecIf("SIP/700-00000085", "1,Congestion") in new stack
  == Spawn extension (emergency, s, 1) exited non-zero on 'SIP/700-00000085'
    -- Executing [h@emergency:1] ExecIf("SIP/700-00000085", ",Set,GLOBAL(EMERGENCY)=3") in new stack
elastix*CLI>
```

Fig 4.2.4: Asterisk CLI demonstrating the process when all 911 emergency calls in progress

As we can see in the above diagram when visitor with extension 700 decided to call since all 911 calls were busy, an IVR response indicating that the line was busy with emergency calls and hence could not disconnect any call –response was initiated.

Here is a case where the successful execution of the emergency call handling application is demonstrated:

The user groups namely Visitor(Extension number:700),Student(Extension number:600),Manager(Extension number:400) and CEO(Extension number:300) are all dialing non-911 extension numbers and all the lines are busy as shown in the below diagram:



Protocol: -none-
Account: sip:300@129.128.209.102

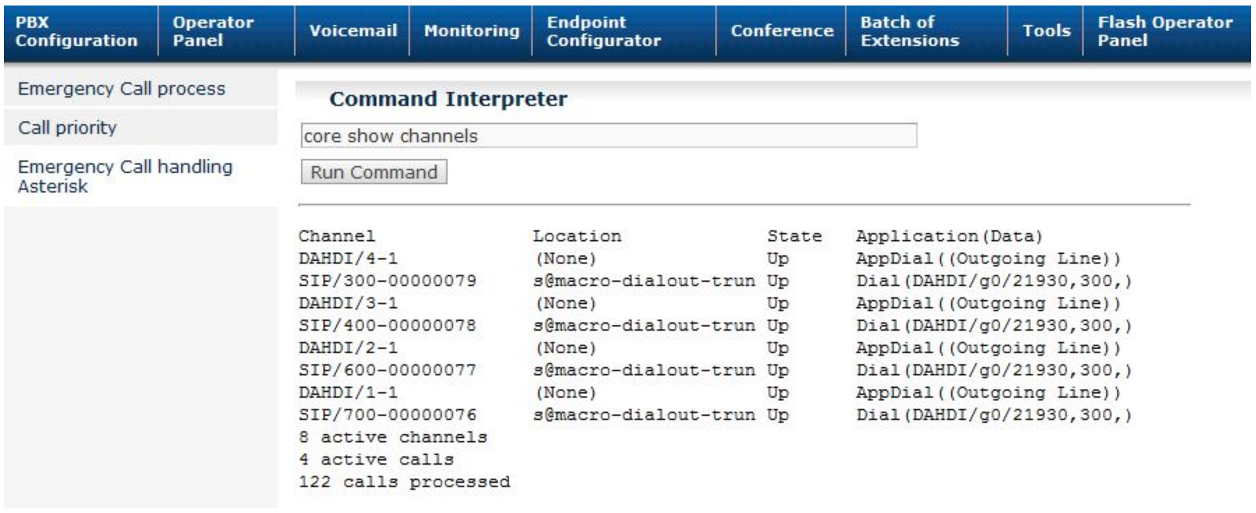
Party	Time	Status
21930	0:52	answered
21930	0:28	answered
21930	0:15	answered
21930	0:07	answered

Call answered: client/48

Fig 4.2.5: All lines dialing non-911 lines and are accepted

So according to the Cost assignment script(context.php)which runs in the macro-dialout trunk the cost values with user groups: CEO has a cost of “700”,Manager has a cost of “600”,Staff has a cost of “500”,Student has a cost of “400”, Visitor has a cost of “300”.

The user groups namely CEO, Manager, Student and Visitor are busy dialing non-911 lines as shown in the below diagram:



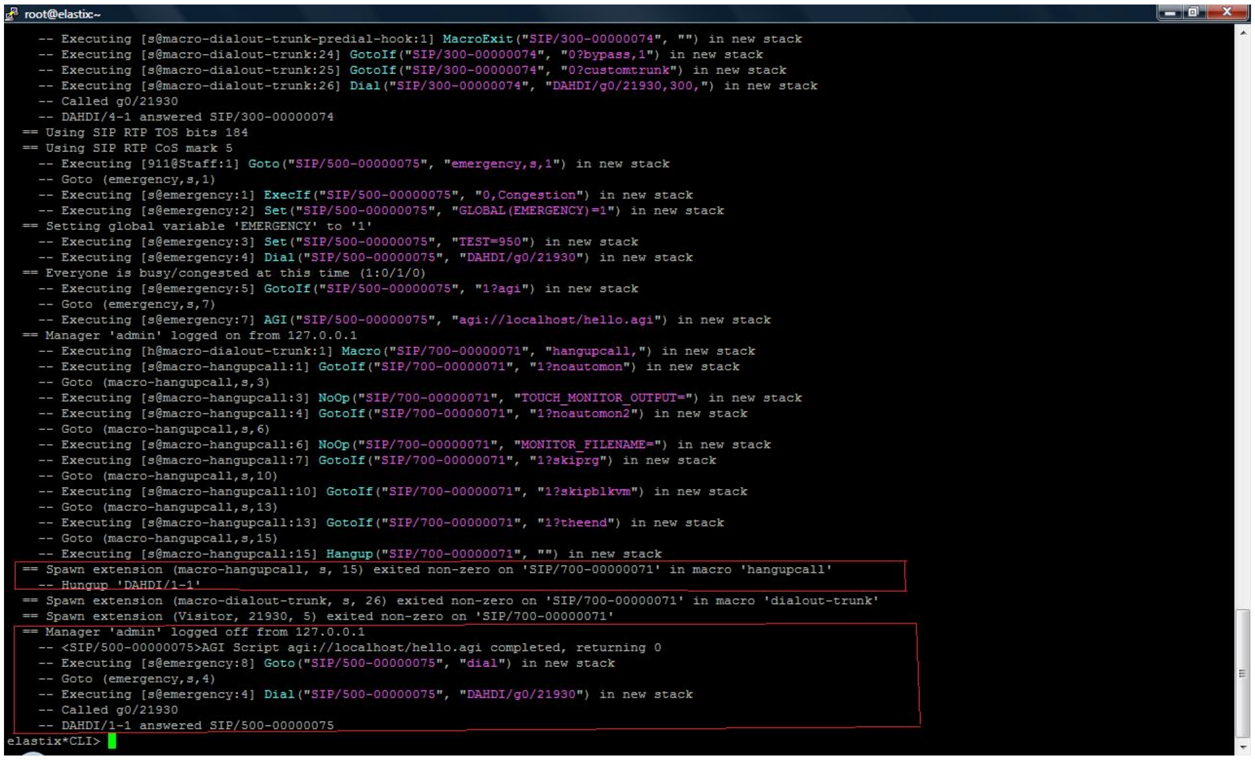
core show channels

```
Channel              Location             State   Application(Data)
DAHDI/4-1            (None)               Up      AppDial((Outgoing Line))
SIP/300-00000079     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
DAHDI/3-1            (None)               Up      AppDial((Outgoing Line))
SIP/400-00000078     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
DAHDI/2-1            (None)               Up      AppDial((Outgoing Line))
SIP/600-00000077     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
DAHDI/1-1            (None)               Up      AppDial((Outgoing Line))
SIP/700-00000076     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
8 active channels
4 active calls
122 calls processed
```

Fig 4.2.6: Elastix GUI demonstrating that all four available channels are busy with non-911 lines

Now, Staff has to dial 911 for emergency in the current scenario but all the lines are busy and there is congestion in the network.

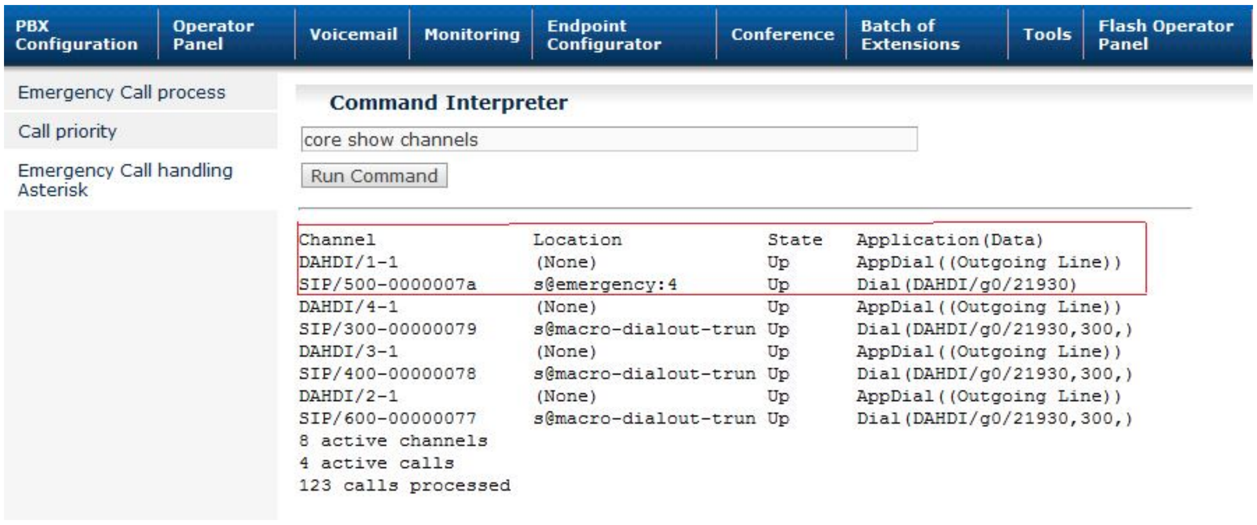
So when the staff dials 911 during this busy period, the DefaultAGIserver that runs in the background maps on to the Emergency Call handling script.The script finds that among all the busy channels the one which is having the lowest cost is Visitor(Extension :700) and its cost is “300”.Hence it will release the channel having the lowest cost that is Visitor. This process is shown in below :



```
  -- Executing [s@macro-dialout-trunk-predial-hook:1] MacroExit("SIP/300-00000074", "") in new stack
  -- Executing [s@macro-dialout-trunk:24] GotoIf("SIP/300-00000074", "0?bypass,1") in new stack
  -- Executing [s@macro-dialout-trunk:25] GotoIf("SIP/300-00000074", "0?customtrunk") in new stack
  -- Executing [s@macro-dialout-trunk:26] Dial("SIP/300-00000074", "DAHDI/g0/21930,300,") in new stack
  -- Called g0/21930
  -- DAHDI/4-1 answered SIP/300-00000074
 == Using SIP RTP TOS bits 184
 == Using SIP RTP CoS mark 5
  -- Executing [911@Staff:1] Goto("SIP/500-00000075", "emergency,s,1") in new stack
  -- Goto (emergency,s,1)
  -- Executing [s@emergency:1] ExecIf("SIP/500-00000075", "0,Congestion") in new stack
  -- Executing [s@emergency:2] Set("SIP/500-00000075", "GLOBAL(EMERGENCY)=1") in new stack
 == Setting global variable 'EMERGENCY' to '1'
  -- Executing [s@emergency:3] Set("SIP/500-00000075", "TEST=950") in new stack
  -- Executing [s@emergency:4] Dial("SIP/500-00000075", "DAHDI/g0/21930") in new stack
 == Everyone is busy/congested at this time (1:0/1/0)
  -- Executing [s@emergency:5] GotoIf("SIP/500-00000075", "1?agi") in new stack
  -- Goto (emergency,s,7)
  -- Executing [s@emergency:7] AGI("SIP/500-00000075", "agi://localhost/hello.agi") in new stack
 == Manager 'admin' logged on from 127.0.0.1
  -- Executing [h@macro-dialout-trunk:1] Macro("SIP/700-00000071", "hangupcall,") in new stack
  -- Executing [s@macro-hangupcall:1] GotoIf("SIP/700-00000071", "1?noautomon") in new stack
  -- Goto (macro-hangupcall,s,3)
  -- Executing [s@macro-hangupcall:3] NoOp("SIP/700-00000071", "TOUCH_MONITOR_OUTPUT=") in new stack
  -- Executing [s@macro-hangupcall:4] GotoIf("SIP/700-00000071", "1?noautomon2") in new stack
  -- Goto (macro-hangupcall,s,6)
  -- Executing [s@macro-hangupcall:6] NoOp("SIP/700-00000071", "MONITOR_FILENAME=") in new stack
  -- Executing [s@macro-hangupcall:7] GotoIf("SIP/700-00000071", "1?skiprg") in new stack
  -- Goto (macro-hangupcall,s,10)
  -- Executing [s@macro-hangupcall:10] GotoIf("SIP/700-00000071", "1?skipblkvm") in new stack
  -- Goto (macro-hangupcall,s,13)
  -- Executing [s@macro-hangupcall:13] GotoIf("SIP/700-00000071", "1?theend") in new stack
  -- Goto (macro-hangupcall,s,15)
  -- Executing [s@macro-hangupcall:15] Hangup("SIP/700-00000071", "") in new stack
 == Spawn extension (macro-hangupcall, s, 15) exited non-zero on 'SIP/700-00000071' in macro 'hangupcall'
  -- Hungup 'DAHDI/1-1'
 == Spawn extension (macro-dialout-trunk, s, 26) exited non-zero on 'SIP/700-00000071' in macro 'dialout-trunk'
 == Spawn extension (Visitor, 21930, 5) exited non-zero on 'SIP/700-00000071'
 == Manager 'admin' logged off from 127.0.0.1
  -- <SIP/500-00000075>AGI Script agi://localhost/hello.agi completed, returning 0
  -- Executing [s@emergency:8] Goto("SIP/500-00000075", "dial") in new stack
  -- Goto (emergency,s,4)
  -- Executing [s@emergency:4] Dial("SIP/500-00000075", "DAHDI/g0/21930") in new stack
  -- Called g0/21930
  -- DAHDI/1-1 answered SIP/500-00000075
elastix*CLI>
```

Fig 4.2.7: The Visitor call having the lowest cost priority is being freed for 911 call dialed by the user group “Staff” in the busy congested network

So the channel having the lowest cost is released and application script successfully runs and it is seen in the Emergency call handling asterisk as shown below:

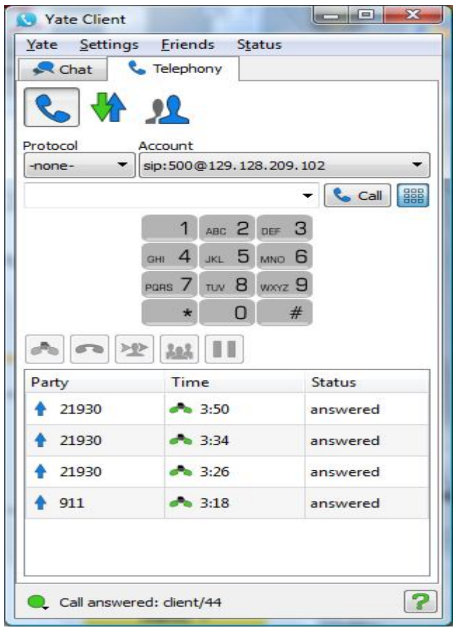


```
core show channels
```

```
Channel              Location             State   Application(Data)
DAHDI/1-1            (None)               Up      AppDial((Outgoing Line))
SIP/500-0000007a     s@emergency:4        Up      Dial(DAHDI/g0/21930)
DAHDI/4-1            (None)               Up      AppDial((Outgoing Line))
SIP/300-00000079     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
DAHDI/3-1            (None)               Up      AppDial((Outgoing Line))
SIP/400-00000078     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
DAHDI/2-1            (None)               Up      AppDial((Outgoing Line))
SIP/600-00000077     s@macro-dialout-trun Up      Dial(DAHDI/g0/21930,300,)
8 active channels
4 active calls
123 calls processed
```

Fig 4.2.8: The Staff user group with Extension “500” is facilitated to attend emergency call after releasing the Visitor call(Extension:700) (which was dialing non-911and call was in progress)

The below diagram also demonstrates successful release of the channel with the lowest cost variable in the Elastix server:



Protocol: -none-
Account: sip:500@129.128.209.102

Party	Time	Status
21930	3:50	answered
21930	3:34	answered
21930	3:26	answered
911	3:18	answered

Call answered: client/44

Fig 4.2.9: VoIP yate client demonstrating that the Visitor call was disconnected

The Emergency call module application works with current four lines being set up and is being tested from the Elastix GUI.

The $DIALSTATUS global variable is currently returns Congestion message if the application has more number of channels or lines available in their set up(>=5).This Elastix GUI was developed based on the number of trunks the application was supporting initially and based on number of lines selected by the previous developer in the first phase of the project. Changes to this requires minor customization in expanding the solution to more number of trunks in the macro-dialout trunk.

The following steps would guide us to expand the current scope to cater it to a wider channel base when the application is deployed by the organizations in the future:

1) The Macro-dialout trunk needs to be modified/changed to enable the $DIALSTATUS variable to change its state to congestion based on the limit fixed(Number of available channels) depending on the number of lines the organization would like to support in their setup.
2) The [emergency] in the extensions.conf needs to be changed to a value that corresponds with the available number of free channels to be used in the application
3) The Emergency call handling script in java needs to be modified suitably so that it executes the action on the events in the Asterisk accordingly depending on number of channels concurrently supported in it.
4) The Elastix GUI needs to upgraded with minor web developments when the number of channels are increased and Elastix can be deployed across a wider range of users and user groups.

4.3 Plug and Play Module implementation of the Emergency Call Handling Application:

The following section explains the process by which the Emergency call handling application can be implemented as plug and play module in a new elastix setup:

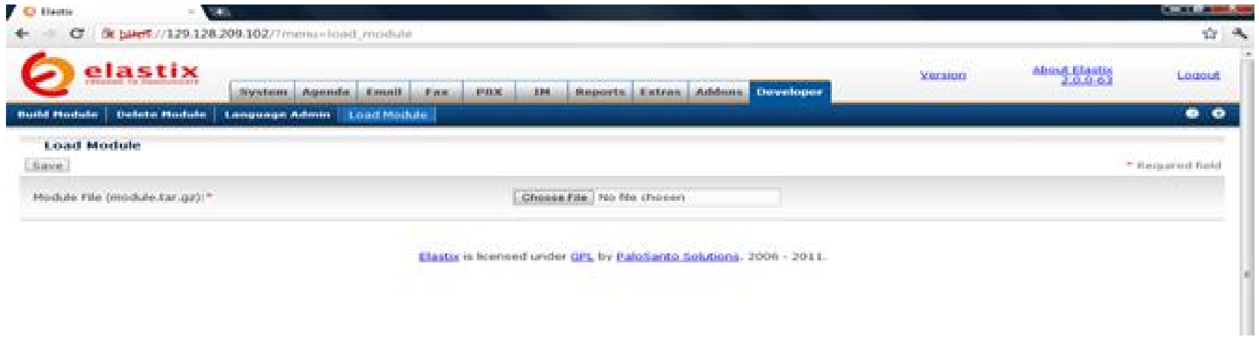
The Elastix Emergency call handling application consists of following modules in the asterisk server :

a) Call priority module
b) Emergency call handling
c) Emergency call handling Asterisk tool

These modules are found in the /var/www/html folder of the Elastix server. The below section explains the process to implement the Emergency call handling application in a new Elastix server:

a) The Developer module needs to be installed in the Elastix web server.

b) The Emergency call handing application module will be loaded in the load module section of the Developer module.



Elastix is licensed under GPL by PaloSanto Solutions. 2006 - 2011.

Fig 4.3.1:Loading the plug & play elastix call handling application module

The module will be loaded in the form of module.tar.gz. Each of the developed modules will be extracted in the form of module_name.tar.gz format.

c) Once the modules are installed ,the elastix server needs to ensure updated version of Java is present in the server. Also to ensure Java compiler support is obtained in the centos of the Elastix server

d) Latest stable version of Asterisk-Java 1.0.0.M3 needs to be installed in the Elastix Asterisk server

e)Open-Source PHP-Java bridge needs to be installed in the server. This is optional and is required to be installed only if the developers anticipate the need to require support for Java classes in their PHP code f)The starting of DefaultAGI server which is the important task that ensures successful running of the Java-Asterisk AGI Algorithm should be started in the background as cron job in the form of a simple shell script that is included in the Emergency call handling application.

g)SIP Clients(softphones) needs to be installed and the accounts of the user groups like CEO,Student,Manager,Visitor etc needs to be created in the SIP VoIP client software

h) The extension.conf in /etc/asterisk/ folder of Elastix server needs to be updated with the following dial-plan :

```
[emergency]
exten => s,1,ExecIf($[${EMERGENCY}>=4],Congestion)
exten => s,n,Set(GLOBAL(EMERGENCY)=$[${EMERGENCY} + 1])
exten => s,n,Set(TEST=950)
exten => s,n(dial),Dial(${EMERGENCY_TRUNK}/${EMERGENCY_NUM})
exten => s,n,GotoIf($["${DIALSTATUS}"="CONGESTION"]?agi)
exten => s,n,Wait(12)
exten => s,n(agi),Agi(agi://localhost/hello.agi)
exten => s,n,Goto(dial)
exten => h,1,ExecIf($[${TEST}=950],Set,GLOBAL(EMERGENCY)=$[${EMERGENCY} - 1])
```

(Here $TEST is a cost variable used only for testing purposes)

```
[CEO]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Manager]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Student]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Staff]
include => from-internal
exten => 911,1,Goto(emergency,s,1)

[Visitor]
include => from-internal
exten => 911,1,Goto(emergency,s,1)
```

The globals_custom.conf needs to be updated with the following information:

EMERGENCY_TRUNK=DAHDI/g0
EMERGENCY_NUM=911(*MINT Lab Extension number 21930 is used as Emergency number for testing purposes*)
EMERGENCY=0

The macro-dialout trunk needs to include the following information after the line
exten => s,1,Set(DIAL_TRUNK=${ARG1}) in the extensions_additional.conf file :

```
exten => s,n(agi),Agi(/var/www/html/context.php)
exten => s,n,NoOp(Returned new priority:${PRIORITY})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "CEO"],Set,COST=${Cost})
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Manager"],Set,COST=$[${Cost}-100])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Staff"],Set,COST=$[${Cost}-200])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Student"],Set,COST=$[${Cost}-300])
exten => s,n,ExecIf($["${MACRO_CONTEXT}" = "Visitor"],Set,COST=$[${Cost}-400])
```

i)The Asterisk should contain all the required Java and PHP files .The application code is in the form of Hello.agi script which in turn calls the script HelloEvents.java .The following files need to be downloaded into the Asterisk server:
a)HelloEvents.java
b)HelloLive.java

A properties file fastagi-mapping that will map the AGI script and the Java code using the line hello.agi=HelloEvents needs to be included in the /etc/asterisk folder of the asterisk server

j)The shell script new.sh needs to be made an entry into the crontab for the application to have its DefaultAGIServer to get started.This shell script contains the commands to set java classpath to map the DefaultAGIServer to accept the incoming connections of the Asterisk events when Emergency call process is in place.

k)Certain PHP files which provide PHP-Agi support which is described below needs to be included in the asterisk server:
a) phpagi-asmanager.php
b) phpagi.php

The file context.html and context.php which was designed with a view to set up context priorities (cost) from the Elastix GUI will be included as part of the Emergency Call handling application module and gets loaded with the module.

l)In the PBX section of the Elastix GUI,the new extensions of user groups like CEO,Visitor,Student,Manager,Staff etc needs to be created in the PBX configuration module found in the Elastix GUI and same login credentials should be used when configuring the VOIP SIP clients

m)Any future development of the modules can only be carried out after installing the Developer module for the Emergency call handling application.

5. Conclusion:

The successful results of different scenarios in testing the Emergency call handling application indicate that the user will have the complete flexibility in operating the application from the Elastix GUI with full control. This also eliminates the need for the user/administrator to perform complex tasks by logging into Asterisk CLI and making changes ,running different commands, tasks like starting and stopping/restarting the server when required etc. This also demonstrates the powerful integration of PHP with both Asterisk-Java framework and Asterisk-PHP scripting. Asterisk uncovers the full flexibility that it offers to design the application by writing external scripts in PHP and integrating it in the PHP-Java combined environment. The Java and PHP are two extremely powerful programming languages whose object oriented features are best utilized when designing and developing the emergency call handling application. Additionally, the presence of open source streaming XML based network protocol Java-PHP bridge also offers much needed support in PHP-Java integration.

The Elastix GUI server has been efficiently designed in PHP and customization of the Elastix server requires not only good programming knowledge of Java and PHP but also the need to understand the complexity involved in Asterisk (AGI) Scripting with Asterisk-Java and Asterisk-PHP integration.

This design offers scalability in terms of adding more modules to the Emergency call handling application as and when required by installing the required developer module. Concurrent dial-outs can be increased and application can be further customized based on the needs of the organizations.

One important feature of this design is to integrate the java work within the Elastix web GUI that enables the implementation of connecting PHP (front end) with a Java virtual machine(in the backend) and combining it with Asterisk AGI Scripting.In this way, the main algorithm which is the Java code will run in the backend and is more secure when compared to PHP.

From a developer’s perspective, this Elastix web server architecture offers enormous amount of challenges in terms of extending the existing design and creating new plug and play modules that successfully integrate into any Elastix server when installed and implemented in it. It provides an opportunity to gain exteremely good amount of knowledge in the areas of Asterisk AGI scripting and its integration with PHP and Java framework and leads to a situation where the developers can seamlessly works towards writing different complex scripts and comeout with more enhanced applications in Elastix that can benefit all the users in the VoIP Telephony industry and also the organizations that rely and understand the importance of Elastix in their office setup.

6. Future Scope of Work:

One of the challenges encountered while developing the Emergency call handling application is embedding of Asterisk AGI scripting inside the PHP scripts. When setting the priority for groups from the Elastix GUI ,the Elastix variable offers very little flexibility to change when the cost AGI script runs .The Elastix variable takes values that are defined in the Macro-dial trunkouts. It poses some architectural and AGI scripting limitations when accepting the values coming in from the other local channel variables. Hence they are currently tested and executed with manually fixed assigned values in the script.

This is an interesting area in Asterisk where the developers in future can perform some more analysis and research and bring in their expertise to overcome the very little limitations that Elastix is currently exhibiting so that Elastix architecture can be made even more powerful with seamless integration of different applications that are developed using wide range of programming and scripting languages like Java, PHP, Perl ,Python etc

Another area is the integration of Elastix PBX configurations with the Emergency call handling applications. At present, Elastix does not offer the flexibility of interconnecting and interrelating different modules across different distributed platforms within the same server. The Developer module only facilitates the development of new modules or addition/deletion of the new and existing modules as separate independent entities into the Elastix web GUI. The interoperability between modules is one such area in Elastix where there is enormous amount of research that needs to be done thereby making Elastix the extremely powerful open source VoIP communications server .

Concurrent dialouts can be increased and application can be customized to cater to a wider range of Asterisk channels. Elastix , being open source offers full support to wide range of customizations and hence scalability of the architecture is well supported and can benefit the organizations in the long run.

BIBLIOGRAPHY

1)Elastix-Basic Fault Finding and Techniques.pdf by Bob Fryer, Blue Packets(ACT Australia)

2)http://circuitid.net/forum/viewtopic.php?t=8&p=208

3)http://www.elastix.org/en/developers/elastix-hello-world.html

4)http://ergonomics.about.com/od/computingergonomics/qt/GUI_benefits.htm

5)http://answers.yahoo.com/question/index?qid=20081013083327AAaEy54

6)http://www.teach-ict.com/gcse/software/userinterface/miniweb/pg6.htm

7)http://www.esds.co.in/forum/f11/how-install-php-java-bridge-3570/

8)http://www.elastix.org/en/component/kunena/117-asterisk-a-programming/38062-a-very-simple-phpagi-example.html

9) http://www.voicetrunking.com/sip-trunk/elastix/

10) http://www.asteriskguru.com/tutorials/cli_cmd_14.html

11) http://examples.oreilly.com/9780596009625/openbook/ch09.pdf (AGI-Asterisk.pdf)

12) a)http://php-java-bridge.sourceforge.net/pjb/

b)http://sourceforge.net/projects/php-java-bridge/

13)Emergency call handling in Open Source PBX by Kirupakaran Pirasath, Edmonton, Alberta