



VoIP Laboratory C

VoIP Billing in a Village Telco

(cc) Creative Commons Share Alike Non-Commercial Attribution 3

The Village Telco is composed by four major components: (1) a mesh potato, that acts as a wireless meshed VoIP client, (2) a billing platform, (3) a monitoring system and a (4) IP/VoIP gateway that provides access to voice and IP services via one of multiple trunk technologies.

This laboratory focuses on the installation of a billing system for the Asterisk open source PBX. The document covers the installation and basic configuration of *a2billing*. A2billing is an open source implementation of a telecommunication billing and added value services platform.¹

A2billing is a LAMP (Linux Apache MySQL(Postgresql) PHP) application that interfaces with Asterisk using both the AMI and AGI interfaces.

This laboratory has been tested using Debian etch, Debian etch and half, Ubuntu 8.04 and Ubuntu 8.10. This laboratory has been tested using a2billing v1.3.4

¹ <http://www.asterisk2billing.org>

Table of Contents

PART 1. HW and SW Requirements	3
PART 2. Installation of a2billing.....	4
STEP 1. Get A2Billing source code.....	4
STEP 2. Prepare the Database.....	5
STEP 3. Edit a2billing configuration file.....	5
STEP 4. Fix permissions, files and folders	6
1. SIP and IAX.....	6
2. Music-on-hold (not mandatory).....	7
3. Sound files.....	8
4. Configure Asterisk Manager.....	8
5. Install The AGI components	9
STEP 5. Install web-based Graphical interfaces	9
PART 3. Configure A2Billing.....	10
STEP 1. Create dialplan.....	10
STEP 2. Providers and trunks.....	11
1. Create provider for outgoing VoIP calls.....	11
2. Create provider for local calls.....	12
STEP 3. Ratecards and callplans.....	13
1. Define callplan.....	14
STEP 4. Create ratecards.....	14
1. Create rates.....	14
2. Add rated to callplans.....	15
STEP 5. Create customers.....	16
PART 4. Configure VOIP Clients.....	17
STEP 1. Retrieve sip parameters.....	17
STEP 2. Configuration of SIP parameters.....	17
1. Dialplan.....	17
2. Create four users (Port 1-4).....	18
3. Create a service provider.....	18
4. Calling Rule.....	20
5. Configure second client.....	20
PART 5. Configure Incoming calls.....	21
STEP 1. Add DID.....	21
STEP 2. Add destinations.....	22
PART 6. Configure external SIP client.....	22
STEP 1. Create a SIP USER in Asterisk.....	22
STEP 2. Configure the ATA.....	23
STEP 3. Configure A Softphone.....	25

1. Install softphone.....	25
2. Configure softphone.....	25
PART 7. Practical exercises.....	26
PART 8. Default passwords and URL.....	26

PART 1. HW AND SW REQUIREMENTS

(Minimum) Hardware requirements:

- 2 IP04
- 1 PAP2
- 2 Analogue phones
- 1 PC
- 4 Ethernet cables

Linux Requirements:

1. Apache with Mysql, PHP5 and GD support
2. PHP5-CLI with PCNT support (php -m)²
3. Mysql
 - phpagi 2.14 included http://phpagi.sourceforge.net/A2Billing_AGI/libs_a2billing/phpagi_2_14
 - jgraph_lib (included in a2billing) A2Billing_UI/Public/jgraph_lib

Pre-required Packages

A2billing requires the packages of a LAMP (PHP5) installation.

To install the necessary packages, run the following commands:

```
#apt-get install libapache2-mod-php5 php5 php5-common
#apt-get install php5-cli php5-mysql mysql-server apache2 php5-gd
#apt-get install openssh-server
```

Asterisk is of course also needed.

```
#apt-get install asterisk
```

Setup up the hardware according to Image 1 (leave the PAP2 ATA for later).

2 Check your php.ini for:
magic_quotes_gpc = Off
register_globals = Off

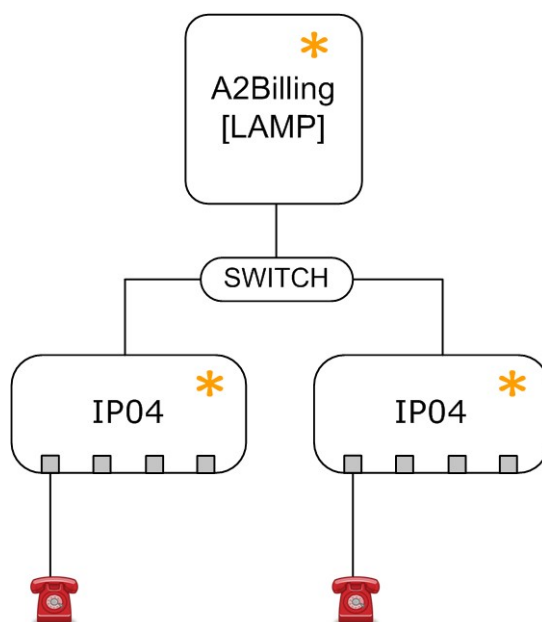


Image 1: Hardware setup for Lab C.

PART 2. INSTALLATION OF A2BILLING

In a nutshell installing a2billing requires seven steps:

1. Download and unpack source code
2. Setup the database
3. Edit a2billing.conf file. Setting up the database parameters
4. Fix permissions and folders
5. Installing the web based graphical user interfaces (Customer and Admin)
6. Place the AGI files
7. Prepare your dialplan

STEP 1. GET A2BILLING SOURCE CODE

We will now create a directory (mkdir) for the software (/usr/local/src/a2billing), download the source code (wget) and unpack it (tar).³

```
#mkdir usr/local/src/a2billing
```

³ If you are a developer consider using the SVN version
svn co --username guest --password guest http://svn.a2billing.net/svn/asterisk2billing/trunk

```
#cd /usr/local/src/a2billing
#wget http://www.asterisk2billing.org/downloads/A2Billing_1.3.4.tar.gz
#tar zxvf A2Billing_1.3.4.tar.gz
```

STEP 2. PREPARE THE DATABASE

We will now create a MySQL database (mya2billing) for the billing software. The file *a2billing-MYSQL-createdb-user.sql* includes a script that creates the database with the correct access control users and permissions.

```
#cd /usr/local/src/a2billing
#mysql -u root -p < DataBase/mysql/Mysql-5.x/a2billing-MYSQL-createdb-
user.sql
```

After creating the database structure, we will create a set of tables and insert some initial basic data that A2Billing needs to function. Again, the instructions on how to create those tables are included in a pre-defined file (*a2billing-mysql-schema-MYSQL.5.X-v1.3.0.sql*).

```
#mysql -u root -p mya2billing < DataBase/mysql/Mysql-5.x/a2billing-
mysql-schema-MYSQL.5.X-v1.3.0.sql
```

Checkpoint 1: Check that the database (my2billing) and that (51) tables have been created.

```
#mysql -u root -p mya2billing
mysql>show tables
mysql>exit
```

STEP 3. EDIT A2BILLING CONFIGURATION FILE

We need to edit the *A2Billing configuration file* (*a2billing.conf* file). The file contains all the configuration information for the Asterisk2Billing platform, such as database connection parameters, AGI behavior and Web Interface customization.

First, we need to copy the default configuration file from the source code (*/usr/local/src*) to Asterisk (*/etc/asterisk*).

```
#cp /usr/local/src/a2billing/a2billing.conf /etc/asterisk/
```

Open the file with your favorite text editor (*vi* is used in this example). If you are new to Linux, we recommend you to use the text editor *Gedit*⁴.

```
#vi /etc/asterisk/a2billing.conf
```

⁴ Gedit is a graphical text editor and can not be run from a console. In that case, we recommend *vi*.

This file contains different sections to configure several parts of the application :

- section [database] : To configure the database connection
- section [webui] : To customize the web user interface
- section [recprocess] : Configuration for the Recurring process (cront job)
- section [agi-confX] : Configuration for the AGI, several configurations can be defined, ie "agi-conf1", "agi-conf2", etc... the groupid parameter will define which configuration to use. Usage : DeadAGI(a2billing.php|%groupid%) by default agi-conf1 is used

Let's have a look at the database parameters in the section [database]. The following example shows the default database configurations.

If you decide to change the database parameters (you should definitely change the password), this section needs to be updated.

Make sure that the port number is correct and that you have the right type of database enabled (mysql).

```
[database]
hostname = localhost
port = 3308
user = a2billinguser
password = a2billing
dbname = mya2billing
;dbtype = postgres
dbtype = mysql
```

STEP 4. FIX PERMISSIONS, FILES AND FOLDERS

In this step, we will tweak the file permissions of Asterisk to fit the A2Billing software. We will also create a number of additional files and folders that A2Billing needs, which does not come with the default installation.

1. SIP and IAX

First we will set a few file permissions (chmod, chown) and create (touch) the SIP and IAX configuration files for Asterisk.

```
chmod 777 /etc/asterisk
touch /etc/asterisk/additional_a2billing_iax.conf
```

```
touch /etc/asterisk/additional_a2billing_sip.conf
echo \#include additional_a2billing_sip.conf >> /etc/asterisk/sip.conf
echo \#include additional_a2billing_iax.conf >> /etc/asterisk/iax.conf
chown -Rf www-data /etc/asterisk/additional_a2billing_iax.conf
chown -Rf www-data /etc/asterisk/additional_a2billing_sip.conf
```

2. Music-on-hold (not mandatory)

Now we will create ten *musiconhold* folders for Asterisk. The folders will look like the following:

```
/var/lib/asterisk/mohmp3/acc_1;
/var/lib/asterisk/mohmp3/acc_2;
/var/lib/asterisk/mohmp3/acc_3;
```

To save you the trouble of creating all the folders with the right permissions, we have created a script that does the job for you. To create the folders, do the following:

Open a text edit and create a file called “create_mohmp3.sh”.

Paste in the code found below, save the file and quit.

```
#!/bin/bash
mkdir /var/lib/asterisk/mohmp3/acc_1
mkdir /var/lib/asterisk/mohmp3/acc_2
mkdir /var/lib/asterisk/mohmp3/acc_3
mkdir /var/lib/asterisk/mohmp3/acc_4
mkdir /var/lib/asterisk/mohmp3/acc_5
mkdir /var/lib/asterisk/mohmp3/acc_6
mkdir /var/lib/asterisk/mohmp3/acc_7
mkdir /var/lib/asterisk/mohmp3/acc_8
mkdir /var/lib/asterisk/mohmp3/acc_9
mkdir /var/lib/asterisk/mohmp3/acc_10
chmod 777 /var/lib/asterisk/mohmp3/acc_*
```

Now, run the script from the command line:

```
#!/bin/bash create_mohmp3.sh
```

Now we need to add *musiconhold classes* to the musiconhold configuration file (*musiconhold.conf*). Again, we have created a script that makes the work for you. Just create a file (*add_mohmp3.sh*) with the code below and run the script from the command line.

```
echo acc_1 =\> mp3:/var/lib/asterisk/mohmp3/acc_1 >>
/etc/asterisk/musiconhold.conf
echo acc_2 =\> mp3:/var/lib/asterisk/mohmp3/acc_2 >>
/etc/asterisk/musiconhold.conf
echo acc_3 =\> mp3:/var/lib/asterisk/mohmp3/acc_3 >>
/etc/asterisk/musiconhold.conf
echo acc_4 =\> mp3:/var/lib/asterisk/mohmp3/acc_4 >>
/etc/asterisk/musiconhold.conf
echo acc_5 =\> mp3:/var/lib/asterisk/mohmp3/acc_5 >>
/etc/asterisk/musiconhold.conf
echo acc_6 =\> mp3:/var/lib/asterisk/mohmp3/acc_6 >>
/etc/asterisk/musiconhold.conf
echo acc_7 =\> mp3:/var/lib/asterisk/mohmp3/acc_7 >>
/etc/asterisk/musiconhold.conf
echo acc_8 =\> mp3:/var/lib/asterisk/mohmp3/acc_8 >>
/etc/asterisk/musiconhold.conf
echo acc_9 =\> mp3:/var/lib/asterisk/mohmp3/acc_9 >>
/etc/asterisk/musiconhold.conf
echo acc_10 =\> mp3:/var/lib/asterisk/mohmp3/acc_10 >>
/etc/asterisk/musiconhold.conf
```

Make sure that the result are ten lines as below (where i=1-10) in the *musiconhold.conf* file.

```
acc_i => mp3:/var/lib/asterisk/mohmp3/acc_i
```

3. Sound files

We need to copy (cp) a few files from A2Billing package to Asterisk sounds folder.

```
cd /usr/local/src/a2billing/addons/
cp sounds/* /usr/share/asterisk/sounds/
cp sounds/en/* /usr/share/asterisk/sounds/
```

4. Configure Asterisk Manager

Configure the Asterisk Manager by editing the *manager.conf* file.

```
#vi /etc/asterisk/manager.conf
[general]
enabled = yes
port = 5038
bindaddr = 0.0.0.0

[myasterisk]
secret=mycode
read=system,call,log,verbose,command,agent,user
write=system,call,log,verbose,command,agent,user
```

5. Install The AGI components

Place the entire content of the directory *A2Billing_AGI* into your *agi-bin* directory.

```
mkdir /usr/share/asterisk/agi-bin
cd /usr/local/src/a2billing/A2Billing_AGI
cp a2billing.php /usr/share/asterisk/agi-bin/
cp -Rf libs_a2billing /usr/share/asterisk/agi-bin/
```

```
Make sure the script is executable
chmod +x /usr/share/asterisk/agi-bin/a2billing.php
```

STEP 5. INSTALL WEB-BASED GRAPHICAL INTERFACES

In this step, we will install the two graphical interfaces of A2Billing, the Administration interface (*A2Billing_UI*) and the Customer interface (*A2BCustomer_UI*).

Place the directories *A2Billing_UI* and *A2BCustomer_UI* into your webserver document root.

```
cp -rf /usr/local/src/a2billing/A2Billing_UI /var/www/
cd /var/www/A2Billing_UI
chmod 777 templates_c

cp -rf /usr/local/src/a2billing/A2BCustomer_UI /var/www/
cd /var/www/A2BCustomer_UI
chmod 777 templates_c
```

Restart your webserver (Apache).

```
/etc/init.d/apache2 restart
```

Checkpoint 2: Direct a browser to the administrative web interface (http://<ip-addr>/A2Billing_UI) and login (check for all default passwords at the end of this document).

Congratulations, you have now completed the software installation part. Let's move on to the configuration!

PART 3. CONFIGURE A2BILLING

Part 3 focuses on configuring the A2Billing software. When we have completed Part 3, we will be able to perform the following operations:

1. A local user can call any other local user and be billed for the phone call
2. Different rates can be applied to different users (Gold and Silver members).
3. A local user can *dial out* (via the VoIP provider) and be billed for the phone call
4. People (outside of the network) can *call in* to any local user.

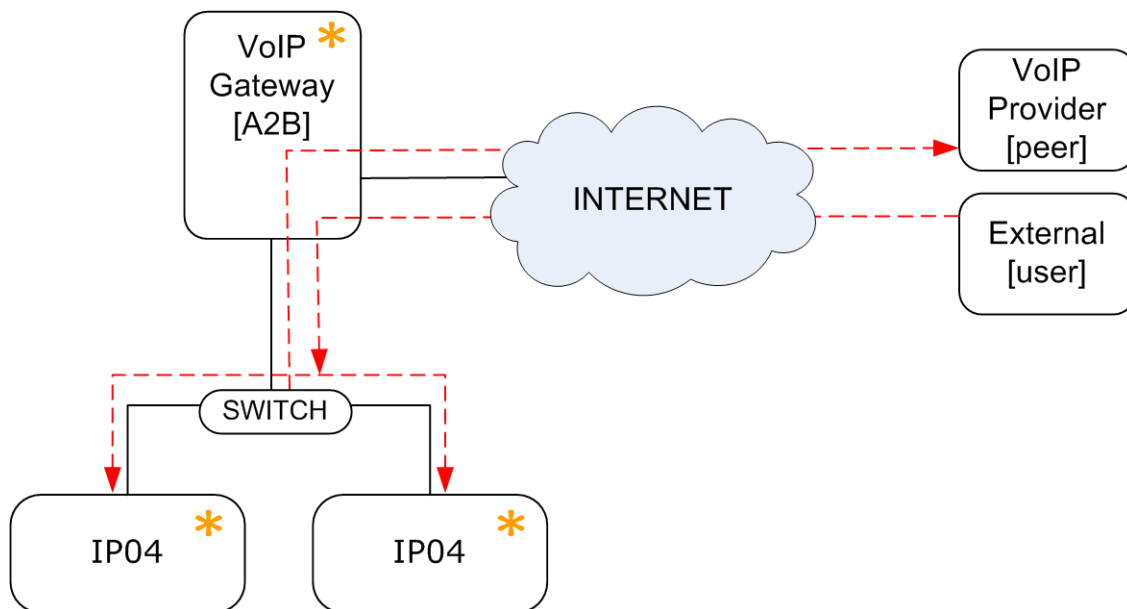


Image 2: The red lines represents voice communication channels between users/peers.

STEP 1. CREATE DIALPLAN

The `extensions.conf` is the Asterisk dialplan. Calls that interact with the billing software need to be handled inside of one or many a2billing related contexts.

The calls that reach the context are processed using the `a2billing.php` AGI script. The `a2billing.php` script can be invoked in many different modes (standard, did, voucher, callback, etc). In the example, we create two different contexts, the first context `[a2billing]` handles all the calls from our VoIP clients. When a call arrives, any extension number `_X`.

(2 digits or more) reaches the script a2billing.php

The second context [did], will be used to route calls back to the users. Calls to the clients (DID) are handled inside of the [did] context. The script a2billing.php in did mode is responsible of routing the call back to one of our users.⁵

Edit *extension.conf* (/etc/asterisk) and add the following two extension.

```
[a2billing]
; CallingCard application
exten => _X.,1,Answer
exten => _X.,2,Wait,2
exten => _X.,3,DeadAGI,a2billing.php
exten => _X.,4,Wait,2
exten => _X.,5,Hangup

[did]
; CallingCard application
exten => _X.,1,Answer
exten => _X.,2,Wait,2
exten => _X.,3,DeadAGI(a2billing.php|1|did)
exten => _X.,4,Wait,2
exten => _X.,5,Hangup
```

STEP 2. PROVIDERS AND TRUNKS

We will now configure A2Billing through its web interface. In our scenario we will create the following providers:

1. The PBX will have one VoIP provider (WA_OUT) to reach the outside world.
 - o **WA_OUT** will use **SIP** as trunk technology
2. The PBX will have one local provider (WA_LOCAL) to reach local clients. The routing of local calls is considered as another provider.

Log into the A2Billing_UI (username/password at the end of this document).

1. Create provider for outgoing VoIP calls

A system can have access to one or more providers to reach the voice world. Each provider can implement one or more voice routing technologies (trunks). We will create one provider/ with one trunk that uses the SIP protocol.

⁵ DeadAGI is a variant of AGI that you use when the channel is hung up.

-
1. Create a provider and name it “WA_OUT” (Trunk>Create Provider)
 2. Create a trunk to the WA_OUT Provider (Trunk>Add trunk)

The trunk should have the following parameters:

VoIP Provider: WA_OUT
Label: VoIP upstreams
Provider tech: SIP
Provider IP: ip-address-of-your-VoIP-provider

Note that if there are any trunks available by default, you can simply delete them.⁶

2. Create provider for local calls

By default, A2Billing does not provide a simple mechanism to route (nor to bill!) clients within the same network, as the default business model behind the software is to provide means to people *to reach other providers*, not to interconnect people on the same network.

Since our main aim is to route phone calls between local clients, and probably also charge for those phone calls, we need to find a work around in A2Billing.

To add the functionality of local routing/billing we need to make sure that all calls to local clients are *processed inside of the DID context*.⁷

1. Create a provider called “WA_LOCAL”.
2. Create a trunk to WA_LOCAL with the following parameters:

VoIP Provider: WA_LOCAL
Label: VoIP local
Provider tech: local
Provider IP: [%dialingnumber%@did](#)

⁶ If the provider requires authentication or special parameters, we will replace the provider IP field for the provider [name]. We will edit manually the file sip.conf and create an entry for the provider [name] where to include the necessary parameters.

⁷ A call to an internal users is going to be forwarded to the [did] local context instead of the VoIP upstream provider.

STEP 3. RATECARDS AND CALLPLANS

This steps handles tariffs and rates for the billing system. The concepts involved, which are important to understand are the following:

Each **client** is associated to **one callplan (a service level)**.

A **call plan** is a collection of *rate cards* connected to a *LC type*.

LC Type refers to an algorithm for route selection. A2Billing supports two different algorithms:

LCR (Least Cost Routing): Select the trunk with the cheapest carrier cost. (buying rate)

LCD (Least Cost Dialing): Select the trunk with the cheapest retail rate (selling rate)

A **rate card** is a collection of call *rates* for certain destinations. The rate card has a fallback trunk/provider defined.

A **rate** includes buying and selling costs (rate/min, connection fee etc.) for a certain trunk/provider.

Image 3 provides an overview on how the different concepts interact with each other.

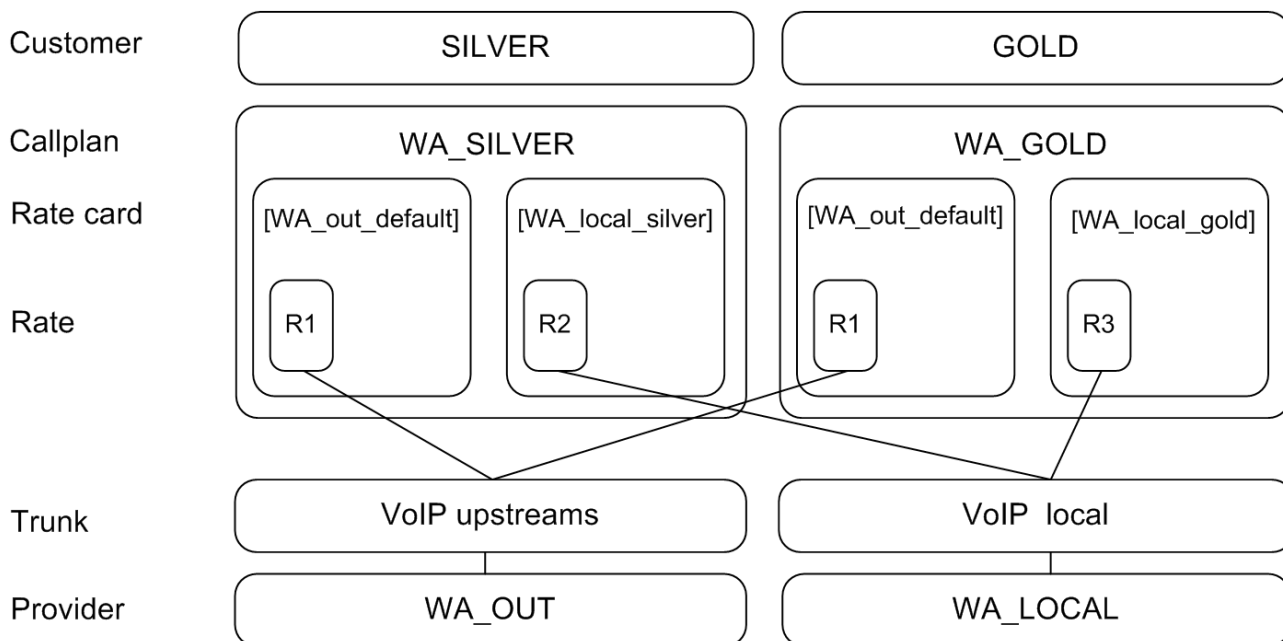


Image 3: Logical overview of A2Billing concepts.

1. Define callplan

We want to be able to charge clients different rates depending on their *member status*. In this example, we will use the member types Gold and Silver.

Gold members enjoys better rates that Silver members.

Create one callplan for the Gold members (WA_GOLD) and one callplan for the Silver members (WA_SILVER). Both callplans should be based on the LCR.

STEP 4. CREATE RATECARDS

Then, create three *rate cards*, one for outgoing calls (both Gold and Silver members), and two for local calls (one for Silver and one for Gold).

Tariff name	Trunk
WA_out_default	VoIP upstreams
WA_local_gold	DID_local_provider
WA_local_silver	DID_local_provider

1. Create rates

Will will now create three different rates, one rate part of each of the rate card. Let's start with the rate for outgoing calls to the VoIP provider.

In this case, we pretend that our VoIP provider routes only calls to Spain, hence the prefix 34.

We have also assumed that the buying rate from the VoIP provider in Spain is 1 USD/minute. Furthermore, we assume that the minimum time the VoIP provider charges for is 30 sec (0.5 USD), and it charges us in blocks of 30 seconds (hence, a phone call of 35 seconds and one of 59 seconds cost the same). Naturally, these figures needs to be adopted to your specific VoIP provider.

In this example, we have decided to sell the outgoing Voip calls for the double rate (2 USD/minute). The minimum time to bill as well as the billing block is set to 60 seconds.

Local phone calls within our network are in fact *free of charge* for the administrator of the network⁸. When we create rates for local calls, we can still choose to charge for those phone calls or not.

⁸ Excluding operation and management of the network.

In this example, we will create two rates for local calls, one rate for Gold members, and one for Silver members. In this way, you can charge clients different rates, depending on their member status.

Parameter	Rate 1	Rate 2	Rate 3
Rate card	WA_out_default	WA_local_gold	WA_local_silver
Dial prefix	34	46	46
Destination	Outgoing	Local gold	Local silver
Buying rate	1	0	0
Buy rate min duration	30	30	30
Buy rate billing block	30	30	30
Selling rate	2	1	1.5
Sellrate min duration	60	30	60
Sellrate min block	60	30	60
Connect charge	0	0	0
Disconnect charge	0	0	0
Trunk	VoIP upstreams	VoIP local	VoIP local

2. Add rated to callplans

Go back to the two callplans you previously created and add suitable ratecards to each one.

Callplan	Ratecard(s)
WA_SILVER	WA_out_default WA_local_silver
WA_GOLD	WA_out_default WA_local_gold

STEP 5. CREATE CUSTOMERS

Customer accounts can be pre-defined in advance, so once a new client joins, one of the idle accounts is handed out to him or her.

We will generate a set of *10 customer accounts* with the following parameters:

Parameter	Value
Length of card number	10
Number of cards	10
Tariff	WA_SILVER
Credit	100 USD
Access	Simultaneously
Currency	USD
Run service	Yes

You have now created 10 customer accounts. In this example, we will only use *the first two accounts*.

Open the first account and view (List Customer, Edit) its contents. As you can see, each customer account is identified with the following unique parameters:

Card number Identifies the user when registering to Asterisk.

Card alias Used as *username* for the A2Billing web login.

WebUI password Password for A2Billing web login.

Change the card alias and webUI password to something easy to remember, as you will use these parameters a number of times.

For example, use 1234 (as card alias and web password) for the first customer, and 5678 for the second customer. Whatever numbers you choose, write them down on a piece of paper.

By default, all customers were created as Silver members. Upgrade the first customer to Gold member, but keep the second customer as Silver Member.

Please note that in this view, you can also edit the customers personal data, such as name, address, email and mobile phone number.

PART 4. CONFIGURE VOIP CLIENTS

We will now configure the two clients of your VoIP network with the user accounts we just created in A2Billing.

STEP 1. RETRIEVE SIP PARAMETERS

Login to the A2Billing customer UI⁹ with the username/password of the first customers you created (the parameters you wrote down).

Go to the SIP/IAX info tab in the main menu. The main window will now display the SIP configuration details for this customer.

The parameters you should pay attention to are the following:

username *same as card number from A2Billing administration UI*

secret *password for SIP registration to Asterisk.*

STEP 2. CONFIGURATION OF SIP PARAMETERS

Login to the Voiptel GUI¹⁰ of your client device.

1. Dialplan

Create a default dialplan (Dialplan 1) under Calling rules.

⁹ http://ip-addr-of-a2billing/A2BCustomer_UI

¹⁰ <Http://ip-addr-of-client-device/>

2. Create four users (Port 1-4)

Parameter	User 1	User 2	User 3	User 4
Extension	1000	2000	3000	4000
Name	1000	2000	3000	4000
Analogue port	1	2	3	4
Dialplan	Dialplan1			
Extension options:	Voicemail, Call waiting, 3-way calling			

3. Create a service provider

All calls from any VoIP client (IP04 or ATA) in the network will go through the VoIP gateway (and A2Billing) before it is routed to its final destination. Also local phone calls (between local clients) will go through the gateway. Depending on the destination of the call, A2Billing will apply different routing and billing rules to each call.

For that purpose, we need to create a *service provider* that connects *your VoIP client (IP04) with the VoIP gateway*.

When we create the service provider we will use the *customer details*¹¹ (*user name, password*) created by A2Billing to identify the user <http://www.voip-info.org/wiki/index.php?page=Asterisk+cmd+DeadAGI>.

As *host* parameter we will state the IP address of the VoIP gateway.

Parameter	Value
Provider type	Customer voip
Comment	WA_OUT
Protocol	SIP
Register	Yes
Host	<ip-address-of-VoIP-gateway>
Username	<A2B-customer-username>
Password	<A2B-customer-password>

If you edit or delete a provider, the Asterisk configuration file *users.conf* sometimes gets corrupted¹². Therefore, after making changes to a provider, it can be a good idea to verify

¹¹ The username/password we retrieved from the A2BCustomer UI for this specific customer.

¹² We know, it should not! But better check it just in case :)

that the *users.conf* file contain the data you wish.

```
#vi /etc/asterisk/users.conf
```

The end of the *users.conf* file (in the IP04) should include the following data:

```
[trunk_1]
disallow = all
allow = alaw,ulaw
callerid =
contact =
qualify = yes
context = DID_trunk_1
dialformat = ${EXTEN:1}
fromdomain = <ip-addr-of-voip-gateway>
fromuser = <A2B card number>
group =
hasexten = no
hasiax = no
hassip = yes
host = <ip-addr-of-voip-gateway>
insecure = very
port = 5060
provider =
registeriax = no
registersip = yes
secret = <A2B-webui-passwd>
trunkname = Custom - WA_OUT
trunkstyle = customvoip
username = <A2B card number>
```

If you edit any Asterisk configuration files manually, do not forget to reload Asterisk with the following command:

```
#asterisk -r
#CLI>reload
```

4. Calling Rule

We need to create a calling rule that routes all our outgoing rules to the provider you just created (WA_OUT).

We will use the prefix 00 to reach the VoIP gateway.

Parameter	Value
Rule name	WA_OUT
Place this call through	Custom WA_OUT
Begins with	00
Follows by	0
Strip	0
Prepend	0

5. Configure second client

Repeat step 1-4 to configure the second VoIP client. Remember to use the username/password from another client account.

Checkpoint 3: Now you should be able to see in your VoIP gateway (that runs Asterisk and A2Billing) that you have two peers registered. Log in to the gateway (ssh) and run the following command:

```
#asterisk -r  
#CLI>sip show peers
```

If you do not see two peers registered, you need to review your configurations.

From the Asterisk command line in the IP04, you can verify that they are registered in the VoIP gateway with the following command:

```
#CLI>sip show registry
```

Checkpoint 4: You should now be able to reach the VoIP gateway by calling 00 from any of your IP04 extensions. To reach any local extension, we still need to add some more configurations.

PART 5. CONFIGURE INCOMING CALLS

We will now configure routing and billing rules for incoming calls to any of our local extensions.

STEP 1. ADD DID

In the A2B web menu, set up billing rules for incoming calls to extension 1000 and 2000 for each IP04. [i] and [j] represents the unique id of your IP04s.

Here we can select if a client should be charged for incoming calls, or if that service should be free of charge. In this scenario, incoming calls will not be charged.

DID	Billing	Monthly rate
46[i]1000	Free	0
46[i]2000	Free	0
46[j]1000	Free	0
46[j]2000	Free	0

Of course, DIDs can be setup for extension 3000 and 4000 as well, but we will only use the first two extensions in this example.

STEP 2. ADD DESTINATIONS

Now, we need to match each DID created with an existing customer and extension.

Create four destinations, one for each extension and customer.

Destination	ID Card	DID
SIP/<cardnumber1>/1000	ID of cardnumber1	46[i]1000
SIP/<cardnumber1>/2000	ID of cardnumber1	46[i]2000
SIP/<cardnumber2>/1000	ID of cardnumber2	46[j]1000
SIP/<cardnumber2>/2000	ID of cardnumber2	46[j]2000

Checkpoint 5: You should now be able to place phone calls between the two IP04s.

Make sure that you can place phone calls *in both directions*.

To call from [i] to [j], do the following:

```
Call 00
Call 46 + [j]+extension number
```

If you can not place phone calls in both directions, you need to review your configurations.

Checkpoint 6: You should not be able to call the VoIP upstreams provider.

```
Call 00
Call 34 + any number
```

The upstream provider in our laboratory will provide you an echo test() no matter which 34<any-number> number you dial.

PART 6. CONFIGURE EXTERNAL SIP CLIENT

Assume that a person in another city, wants to reach one of the clients of the local VoIP network. There are many different ways to do that, through the Internet (IP), the fix telephony network (PSTN) or through GSM. In this example, we will assume that this person is connected to the Internet and is calling you over the IP protocol.

The person calling in, needs either to have a *softphone* installed in his computer, or and *ATA* (and an analogue phone) We will setup both alternatives.

STEP 1. CREATE A SIP USER IN ASTERISK

Create a SIP user, by adding the following code to the end of the file *sip.conf*.

```
[666]
type=friend
username=666
secret=666
disallow=all
allow=ulaw
host=dynamic
qualify=yes
context=fromoutside
```

Then, add the following lines to *extensions.conf*.

```
[fromoutside]
include => did
```

By including did in the context [fromoutside] we are forcing that all calls arriving from the outside via the 666 user can be routed to a DID.

Reload Asterisk

```
#asterisk -r
#CLI>reload
```

STEP 2. CONFIGURE THE ATA

Connect the ATA to the switch and connect an analogue phone to it.

Enable dhcp on your ATA and find out which IP address that was given to it.
Direct your browser to the web server of the ATA.

Action	Command	Choices
Enter configuration menu	****	
Check DHCP	100	
Enable/Disable DHCP	101	Enable: 1 Disable: 0
Check IP address	110	

Configure the ATA to register to the SIP user previously defined (username=666, password=666).

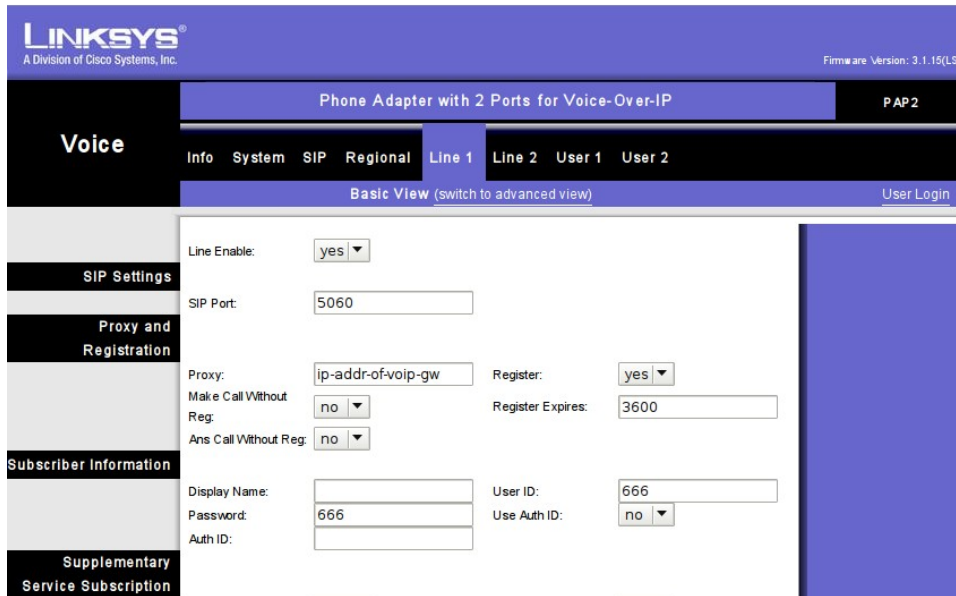


Image 4: Configuration menu of Linksys PAP2T.

Hint: The configuration menu is found under Admin Login, Line 1.

Checkpoint 7: Confirm that the ATA has been registered to the VoIP gateway (there should be three users online now).

Checkpoint 8 : Place a call from the ATA to one of the local extensions.

STEP 3. CONFIGURE A SOFTPHONE

Please note that this example is voluntarily, and requires access to *one more computer*.

1. Install softphone

Install the softphone ekiga in one of your personal computers.

```
#apt-get install ekiga
```

2. Configure softphone

Launch the application and follow the installation wizard. Do not forget to do the audio test!

Create a new account, with the same parameters as we used for the ATA setup.

Checkpoint 9: Call in to any of your IP04s. Does the audio work both ways?



Image 5: UI for the softphone Ekiga.

PART 7. PRACTICAL EXERCISES

The best way to learn more about your VoIP-billing system is to play around with it. Modify rates, billing schemes, credit and other billing related parameters, and see how the system behaves.

Select a couple of the exercises below that you find interesting.

1. Make sure that a customer runs out of credit during a phone call. What happens?
2. Upgrade a customer from Silver to Gold member. Verify that her rates changes.
3. Consider what rates you would charge for local phone calls. What billing parameters would you use?
4. Call to your self using your own DID. What happens?
5. Place two calls simultaneously from the same IP04. What happens to your credit?

PART 8. DEFAULT PASSWORDS AND URL

Type	Username	Password	Database	URL
A2Billing_UI	root	myroot		A2Billing_UI
A2BCustomer_UI	admin	mypassword		A2BCustomer_UI
Manager	myasterisk	mycode		
MySQL database	a2billinguser	a2billing	mya2billing	
IP04 (ssh)	root	uClinux		
VoIPtel GUI	admin	mysecret		

Note! The first character in all usernames and passwords is a small letter (not capital).