



**PC Best Networks SIP PBX Reference
Setup and Development Guide**
(For V2 and V3)

Copyright 2007-2011 PC Best Networks Inc.
support@pcbest.net

Index:

1	Introduction.....	3
2	Installing PBX.....	6
3	PBX Quick Setup Guide	28
3.1	Common Settings	28
3.2	Auto Attendant	32
3.3	ACD(Automatical Call Distribution)	36
3.4	Outbound Calls.....	38
3.5	Dial Extension.....	39
3.6	Virtual Extension.....	41
3.7	Ring group.....	42
3.8	Call Parking.....	44
3.9	Magic Transferring Code (ONLY V3)	46
3.10	FXO/FXS or Digital Gateway.....	47
3.11	Conference Room.....	49
3.12	Inbound 2 Outbound	50
3.13	Setup a music server.....	52
3.14	Echo Test for IP extension.....	53
4	PBX Advanced Call Center Features	54
4.1	Setting up ACD agents.....	54
4.2	Enabling Call Recording.....	57
4.3	Supervisor Call Monitoring	58
4.4	Pickup Group	59
5	PBX Auto Dialer Feature (Pro Only)	61
6	PBX Other Configurations.....	66
6.1	CDR	66
6.2	Networks	67
6.3	System Options	71
6.4	Folders and Logs.....	75
7	PBX Developments (Pro Only).....	76
7.1	Plugin	76
7.2	Manager Client Application (V3 only).....	87
7.3	Database Development (V3)	102
8	Session Border Controller (SBC).....	103

1 Introduction

PC Best Networks provides NO.1 Windows VOIP development kits to business customers. **PC Best IP-PBX** is a proprietary, Windows-based PBX system developed as a response to the growing needs of businesses who want to deploy voice-over-the-internet through a simple, easy to manage platform. There is no difference in the use of **PC Best IP-PBX** whether you are a one-person business or a company with tens or hundreds of staff. Powerful, flexible, light and user-friendly, **PC Best IP-PBX** can be set up and run within 30 minutes on any of your working computer, with great features like, Auto Attendant, ACD(Automatic Call Distribution), MOH(Message On Hold), Ring Group, Call Parking, Pickup Group, Conference, Auto-Dialer, Database Reports, and Plug-in.

Traditional analogue PBX (private branch exchange) solutions have always been out of reach of most small and medium size businesses. Within the last 5 years, the arrival of VoIP phone systems as well as open-source solutions, such as Asterisk, which run on Linux, have become increasingly popular. Today, powerful IP-PBX system can be deployed at a much lower cost than what available 3 or 5 years ago.

Unlike Linux-based programs which may intimidate those who do not have the required expertise or resource to manage, **PC Best IP-PBX** is a user-friendly, Windows-based system and is based on SIP standard that can be set up with little effort by anyone who can configure simple mail programs like Outlook.

PC Best IP-PBX system lets even the smallest businesses quickly employ its rich features and revolutionize day-to-day business's communications. Here are the fundamental business objectives from which **PC Best IP-PBX** was built:

Increase Productivity

By removing the needs for an operator to accept incoming calls, you and/or your front office staff would be able to continue with other workloads. **PC Best IP-PBX's** digital receptionist and extension management features can be set up to answer and transfer the call as how you want.

Save time

PC Best IP-PBX's auto attendant and MOH (Message On Hold) features allow you to provide information about your business that may be relevant to callers' reason for calling you while they are on hold, thus save your time and save your customer's time. Reduce a considerable amount on time spent on the phone with these great features.

Save Cost

PC Best IP-PBX has been built to simply provide just what you want in a PBX system. We keep the development cost low and pass these savings on to you in the form of low initial investment, rather than building a complex system at higher cost with features that you may not need.

Enhance business image

Gone are the days when PBX systems were only suitable for big companies. No matter how small your company may be, your business deserves an image which big companies expose them. By using **PC Best IP-PBX** system, you give your customers a feeling that they are dealing with a well-established organization, thus enhance their confidence.

Improved Customer Services

You and/or your staff will never miss a call, no matter where you are in the world. Whether you're interstate or overseas, **PC Best IP-PBX** can be set up to connect the call to you on fixed line or mobile phone at a cost that is 5 to 10 times lower than call diversion provided by regular telephone networks. Imagine how frustrating your customer might be for not being able to get hold of you. You may be using telephone answering service but other than taking messages for you, these services are limited in what they can do for your business and your customers.

PC Best IP-PBX FEATURES

- Call Logging
- Call Reporting
- Blind Call Transfer
- Attended Call Transfer
- Call Forward on Busy
- Call Forward on No Answer
- Call Routing (DID)
- Conference Calling
- ACD (Hunt Group)
- Auto Attendant / Digital Receptionist
- Voice Mail
- Music On Hold
- Call Parking
- Call Pick Up
- Call Queue
- Call Recording
- Support Plug-in (Customized IVR Menu)

Unified Communications and Mobility

Receive Voice Mail via Email
Public SIP ID for Extensions
Advanced forwarding rules

Supported Codec (Voice Compression)

G711 (a law and u law)
G726-32
GSM
Speex
iLBC
G729

System configuration and call management can be changed instantly and inexpensively via software, not by plugging in circuit cards and pulling cables.

REQUIREMENT:

- Broadband connection
- VoIP service account
- FXO Adapter (optional)
- Minimum Pentium III with 512MB RAM, Windows XP or Vista

Our contact information for support:

Email: support@pcbest.net

Toll Free(USA & Canada): 1-888-733-6620

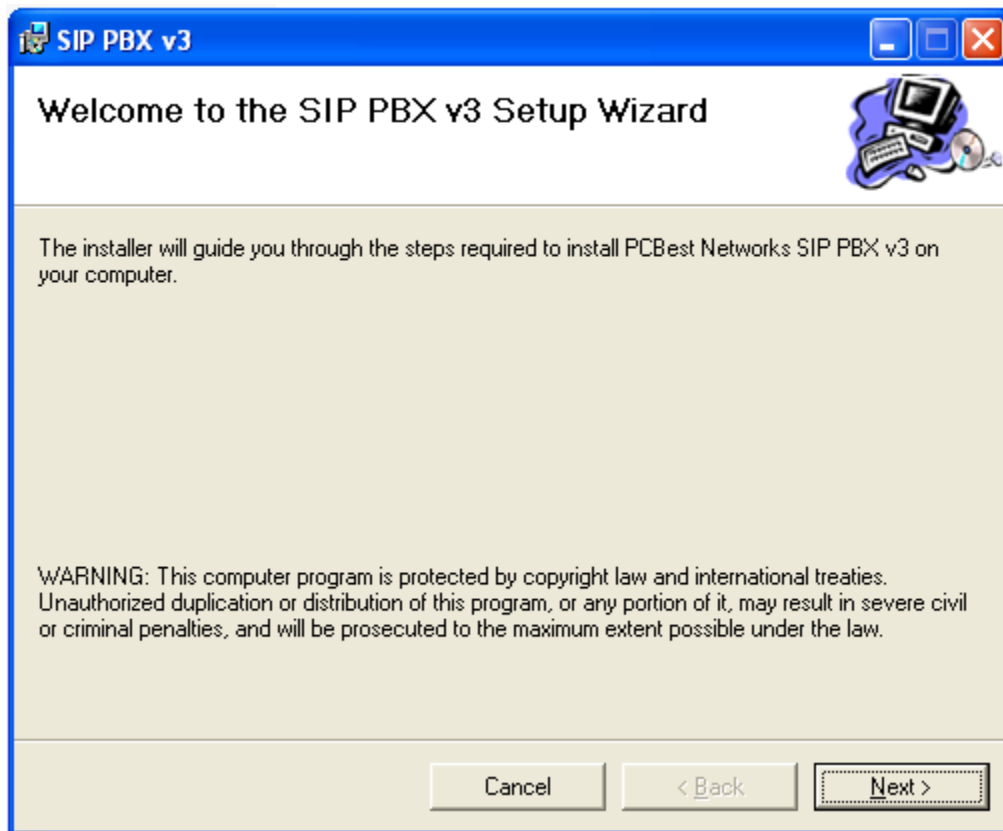
Local and International: 1-613-800-2202

2 Installing PBX

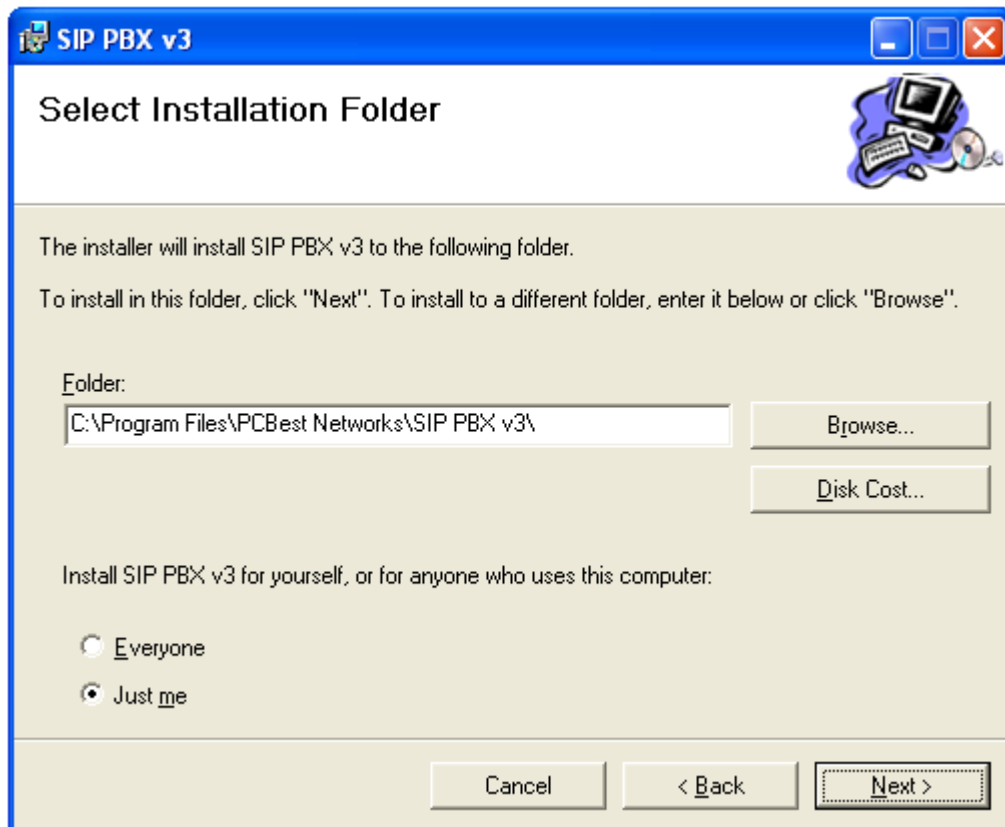
1. **Download** PC Best PBX v3 from this page: http://www.pcbest.net/sip_pbx.php
2. **Unzip** the zip file into a folder. You will see two files:

Name	Size	Type	Date Modified
PCBxv3Setup.msi	7,459 KB	Windows Installer P...	4/6/2010 11:56 AM
setup.exe	421 KB	Application	4/6/2010 11:56 AM

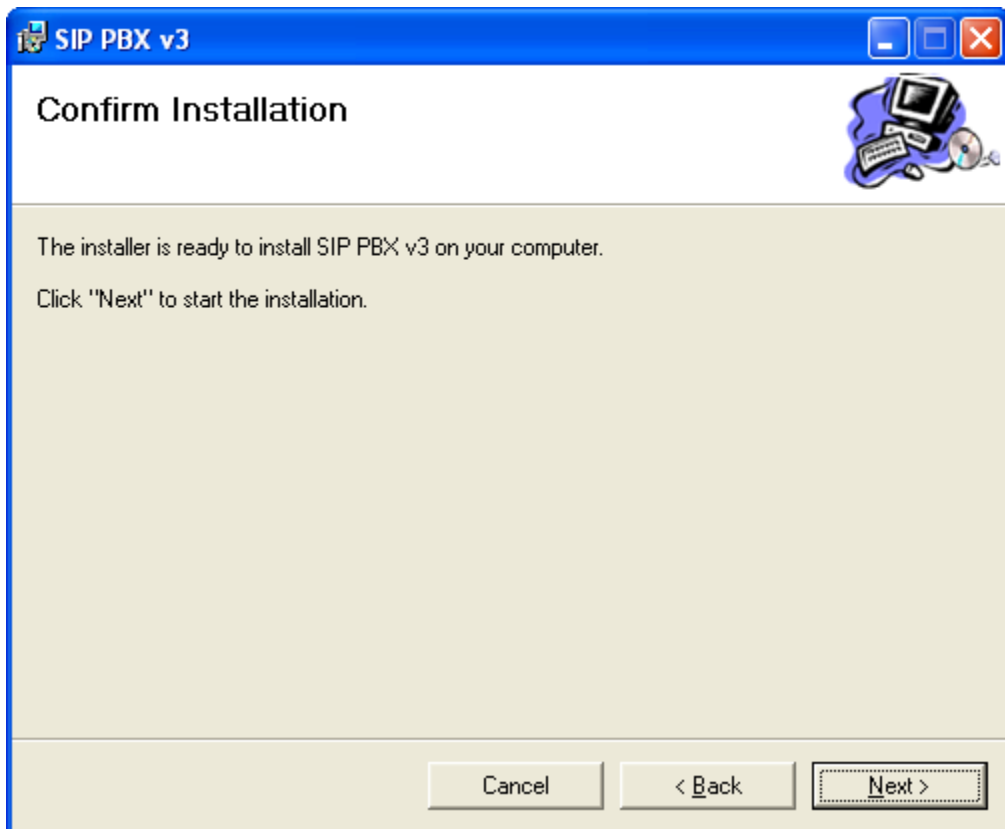
3. Run **setup.exe**. Under Window7 or 2008, please right click **setup.exe**, and **run as administrator**.



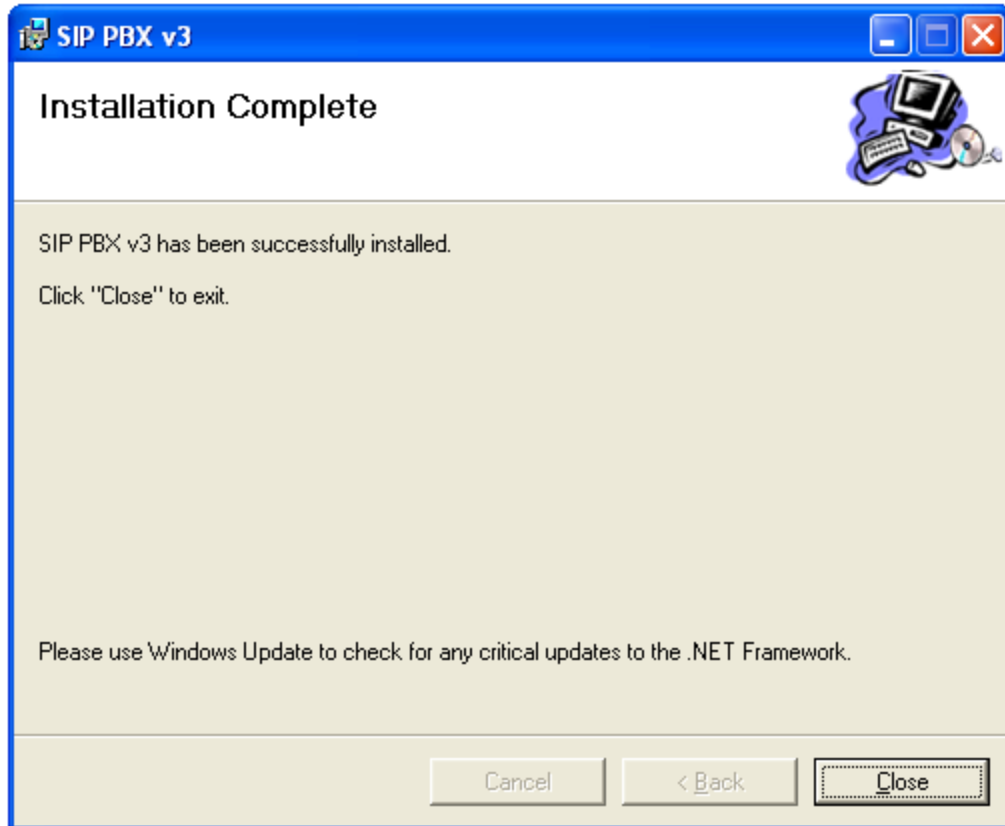
Click next.



Choose where you want to install the program, and who can access it.



Then confirm the installation.



It is done.

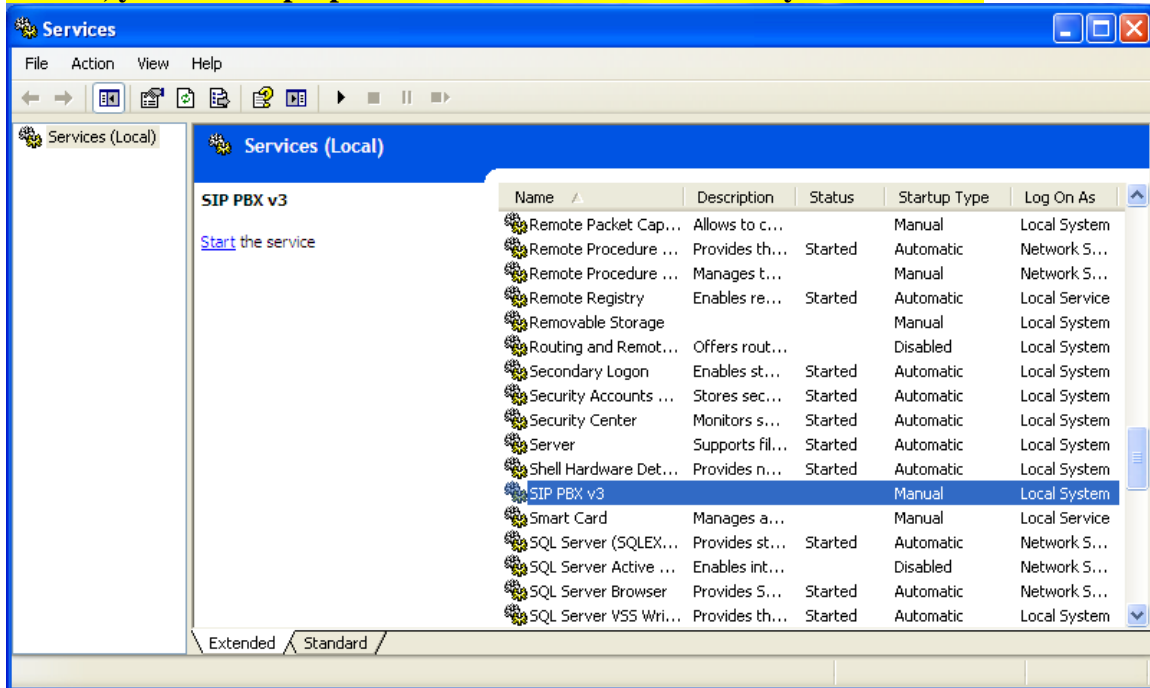
Some customers reported they encountered error 2869, and installation cannot be completed. For this error, it is because of the errors in your Windows registry. There are several ways to solve it(choose one of them):

- a. The 2869 error is a common error regarding MSI files when they are executed in Windows due to the user access control. A workaround for this error is as follows:
 1. Run a command line as an administrator. This can be done by clicking on the start menu and typing CMD . When the option appears, right click and select Run as Administrator.
 2. Run “msiexec /i PBXv3Setup.msi” in the command line, in the directory of two installation files:

Name	Size	Type	Date Modified
PBXv3Setup.msi	7,459 KB	Windows Installer P...	4/6/2010 11:56 AM
setup.exe	421 KB	Application	4/6/2010 11:56 AM

- b. Under Windows 7, 2008, or Vista, please right click setup.exe, and **run as administrator**.

For V3, you need to prepare database for PBX v3 before you can run it.



The SIP PBX v3 service should be in the Windows service list.

For V2, you don't have to setup database in order to run.

V2 is NOT a service application, so you won't see it in Service list like above picture.

4. Setup Database.

Microsoft SQL Server 2005 Express Edition Service Pack 4:

<http://www.microsoft.com/en-ca/download/details.aspx?id=184>

Please download SQLEXPRESS_TOOLKIT.EXE(224.6MB) or
MBSQLEXPRESS_ADV.EXE(254.6 MB).

Microsoft® SQL Server® 2008 Express with Tools:

<http://www.microsoft.com/en-ca/download/details.aspx?id=22973>

Microsoft SQL Server 2008 R2 RTM - Express with Management Tools:

<http://www.microsoft.com/en-ca/download/details.aspx?id=23650>

Microsoft® SQL Server® 2012 Express:

<http://www.microsoft.com/en-ca/download/details.aspx?id=29062>

32bit OS download one of the following:

ENU\x86\SQLEXPRESSADV_x86_ENU.exe 1.3 GB Download

ENU\x86\SQLEXPRESSWT_x86_ENU.exe 706.1 MB Download

64bit OS download one of the following:

ENU\x64\SQLEXPRADV_x64_ENU.exe 1.3 GB Download

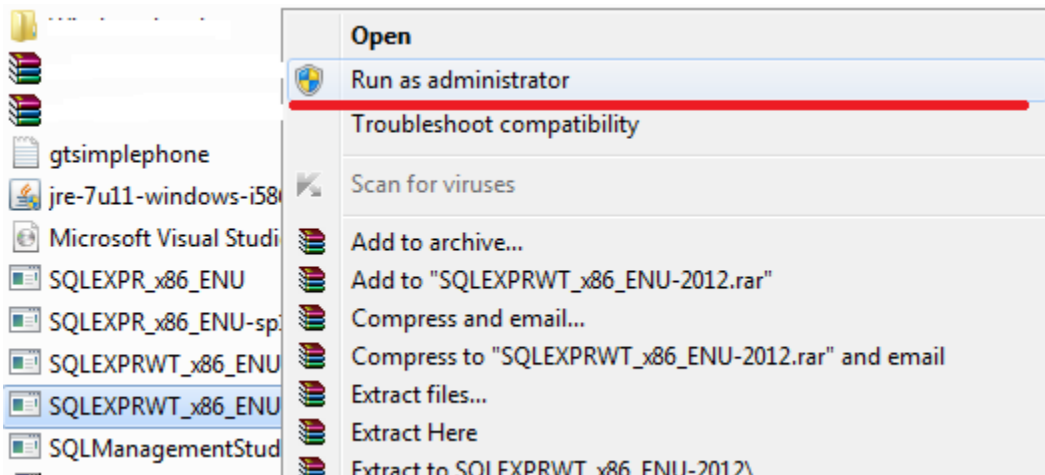
ENU\x64\SQLEXPRWT_x64_ENU.exe 669.9 MB

Assume we use SQL Server 2012 Express here. It is free to download from website.

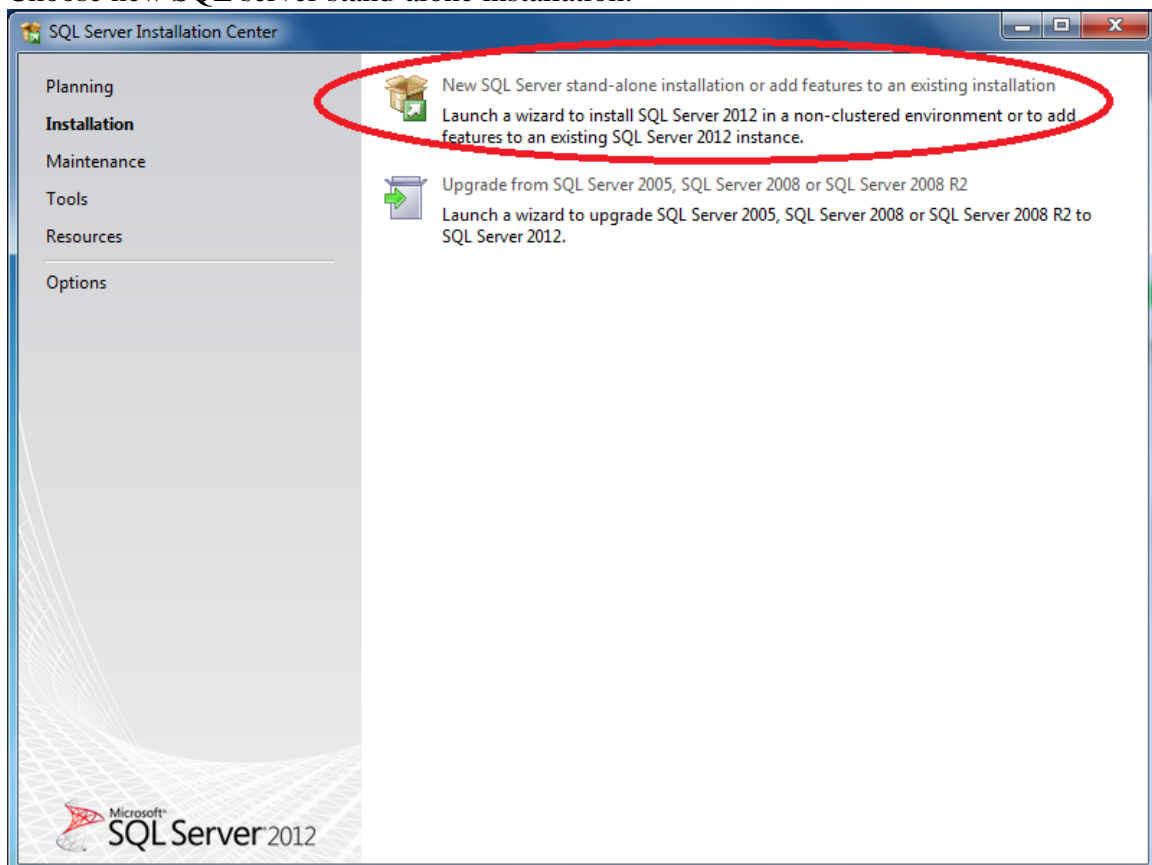
We download SQL Server 2012 with tool, which has management studio.

Right click on SQLEXPRWT_x86_ENU.exe for 32bit Windows or

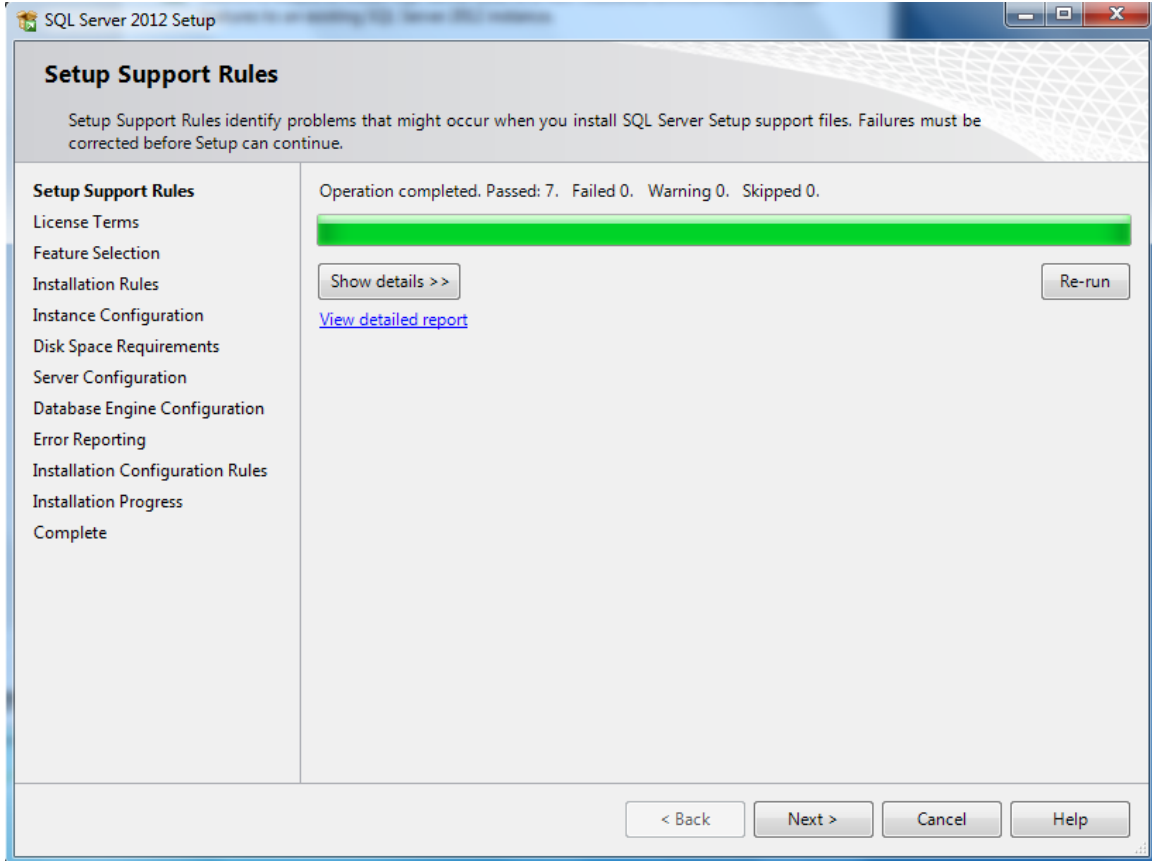
SQLEXPRWT_x64_ENU.exe for 64bit Windows, and "Run as administrator":



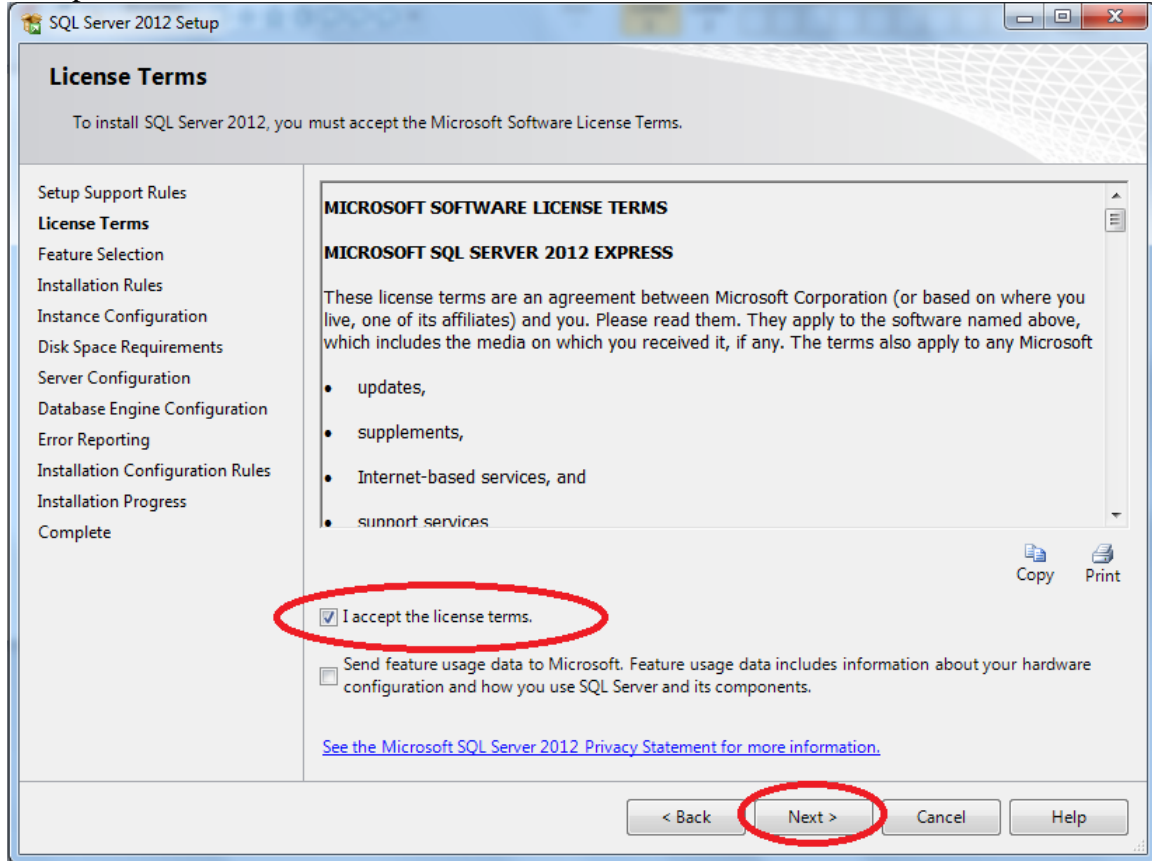
Choose new SQL server stand-alone installation:

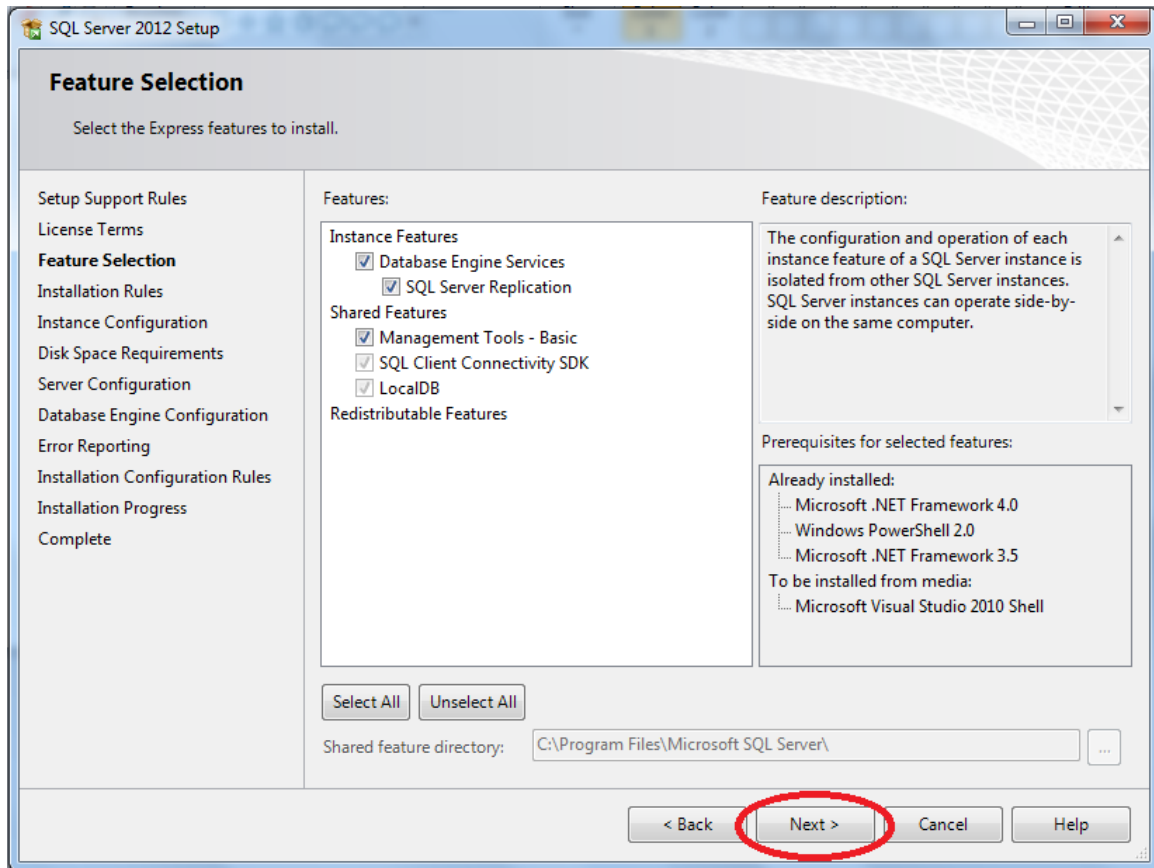


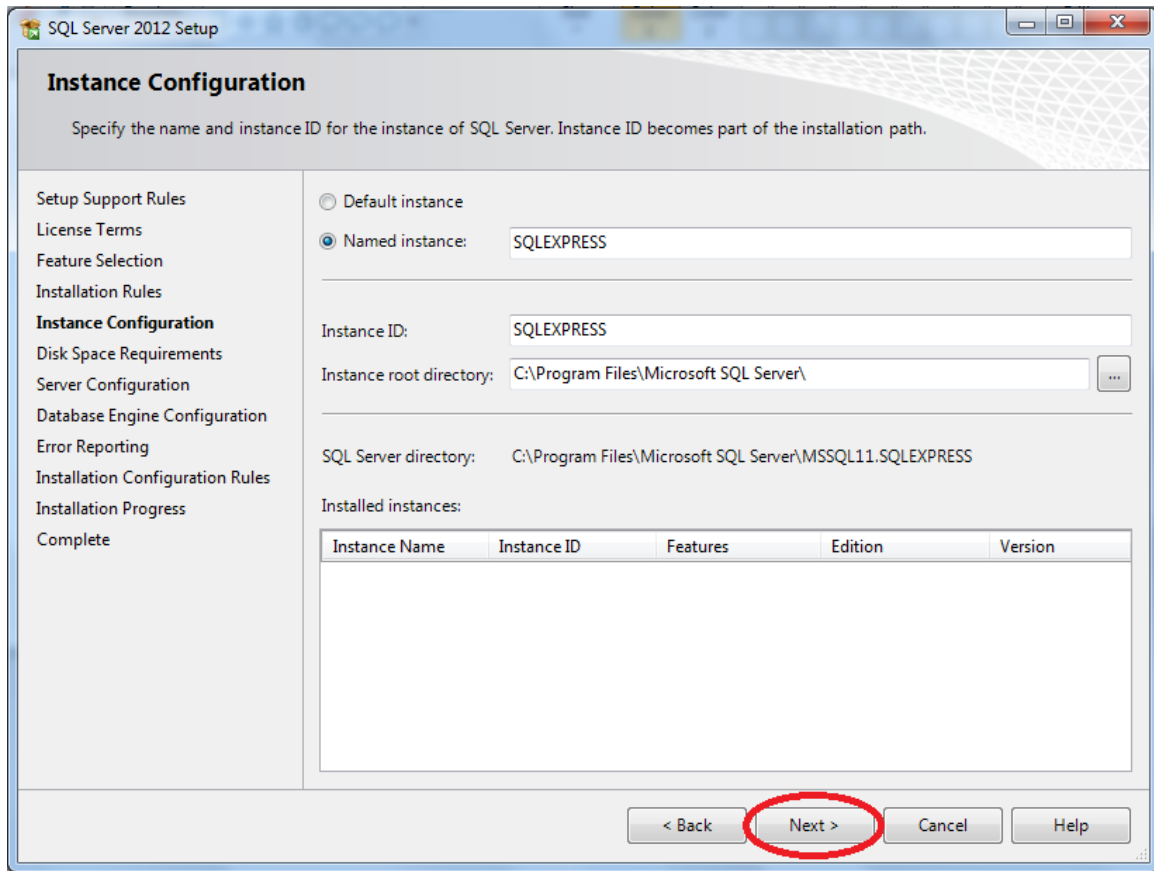
Of course, if you already have 2005, 2008, or 2008R2, you can upgrade it to 2012.
Click next:

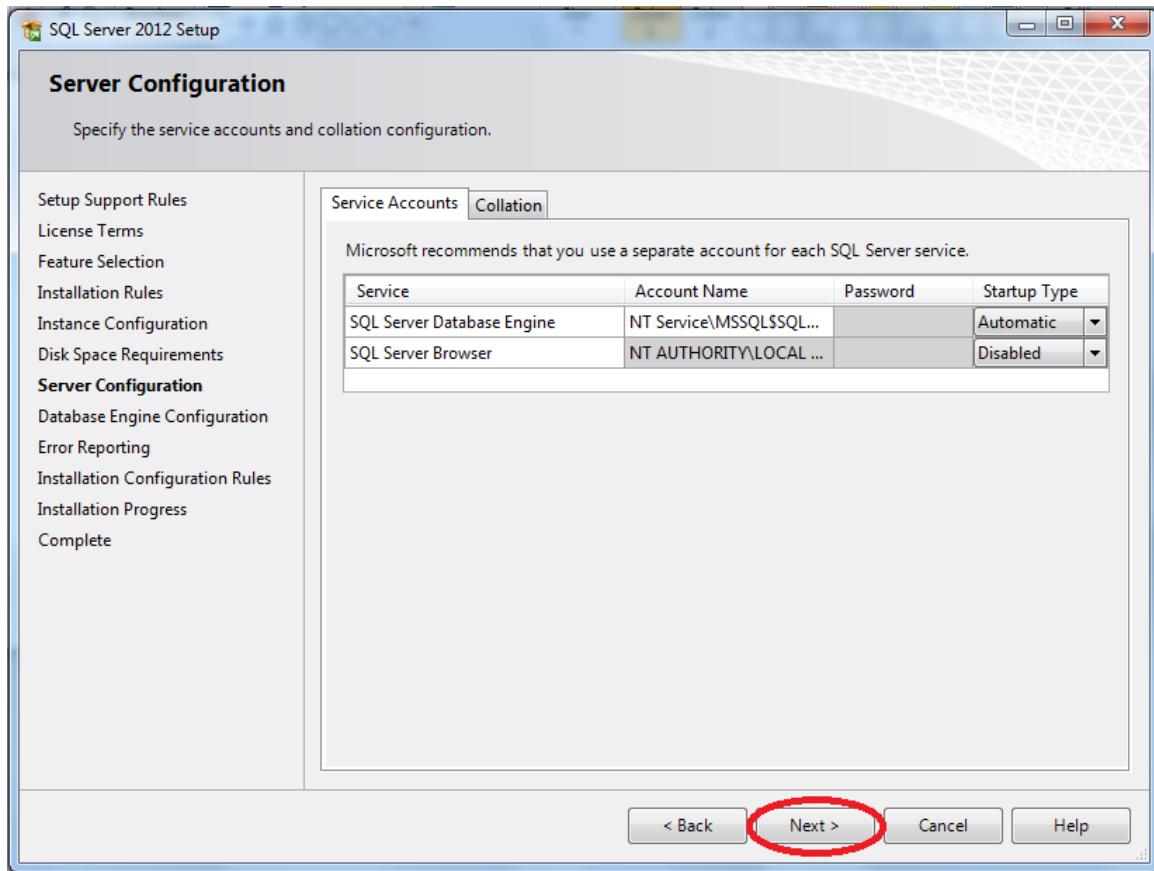


Accept license terms, and click Next:









Choose Mixed Mode, and set password for account sa. NOTE: please write down your password in paper for later use.

The screenshot shows the 'Database Engine Configuration' window in the SQL Server 2012 Setup. The window has a title bar 'SQL Server 2012 Setup' and a subtitle 'Specify Database Engine authentication security mode, administrators and data directories.' On the left is a navigation pane with the following items: 'Setup Support Rules', 'License Terms', 'Feature Selection', 'Installation Rules', 'Instance Configuration', 'Disk Space Requirements', 'Server Configuration', 'Database Engine Configuration' (which is highlighted), 'Error Reporting', 'Installation Configuration Rules', 'Installation Progress', and 'Complete'. The main area has four tabs: 'Server Configuration' (selected), 'Data Directories', 'User Instances', and 'FILESTREAM'. Under the 'Server Configuration' tab, the instruction is 'Specify the authentication mode and administrators for the Database Engine.' The 'Authentication Mode' section has two radio buttons: 'Windows authentication mode' and 'Mixed Mode (SQL Server authentication and Windows authentication)'. The 'Mixed Mode' option is selected and highlighted with a red line. Below this, the instruction is 'Specify the password for the SQL Server system administrator (sa) account.' There are two password fields: 'Enter password:' and 'Confirm password:', both containing masked characters. Below the password fields is a section 'Specify SQL Server administrators' with a list box containing 'Yonge-PCW7\Yonge (Yonge)'. To the right of the list box is a text box that says 'SQL Server administrators have unrestricted access to the Database Engine.' At the bottom of the list box are three buttons: 'Add Current User', 'Add...', and 'Remove'. At the bottom of the window are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

SQL Server 2012 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

Setup Support Rules
License Terms
Feature Selection
Installation Rules
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error Reporting
Installation Configuration Rules
Installation Progress
Complete

Server Configuration | Data Directories | User Instances | FILESTREAM

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

☐ Windows authentication mode

☒ Mixed Mode (SQL Server authentication and Windows authentication)

Specify the password for the SQL Server system administrator (sa) account.

Enter password:

Confirm password:

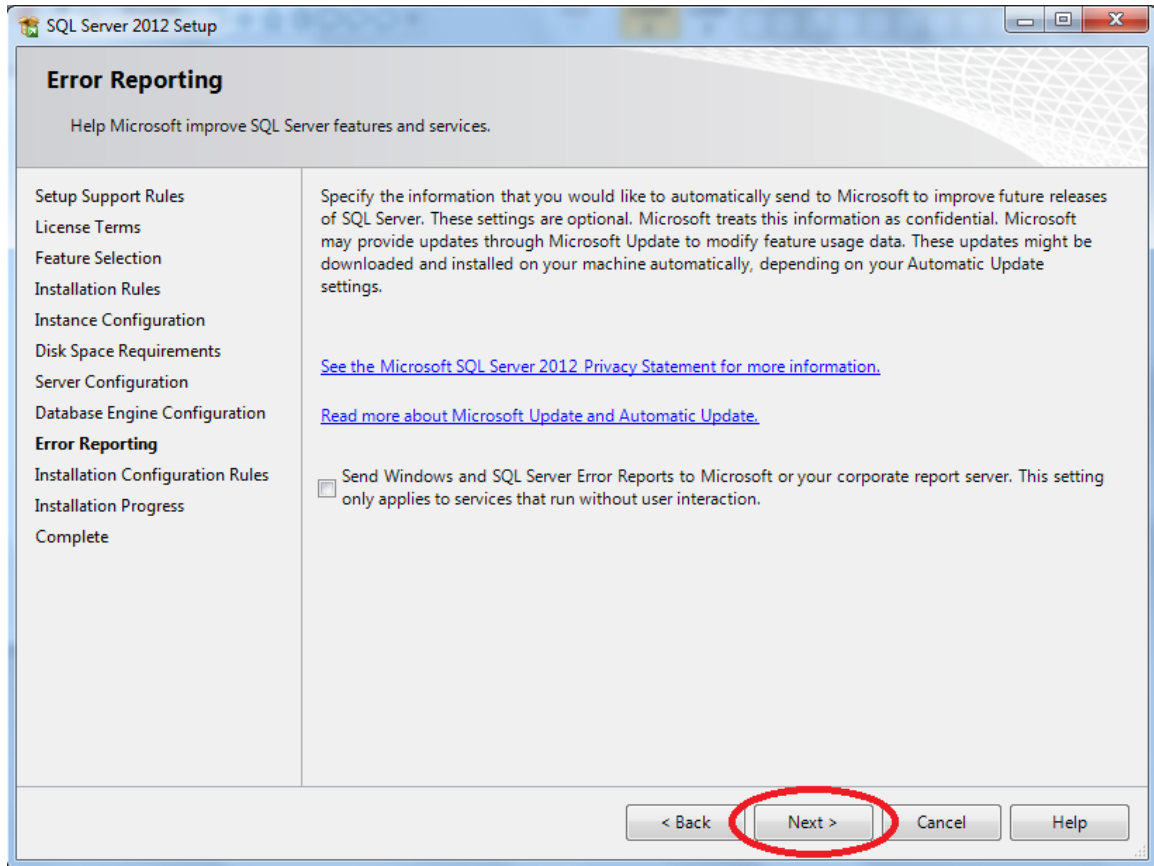
Specify SQL Server administrators

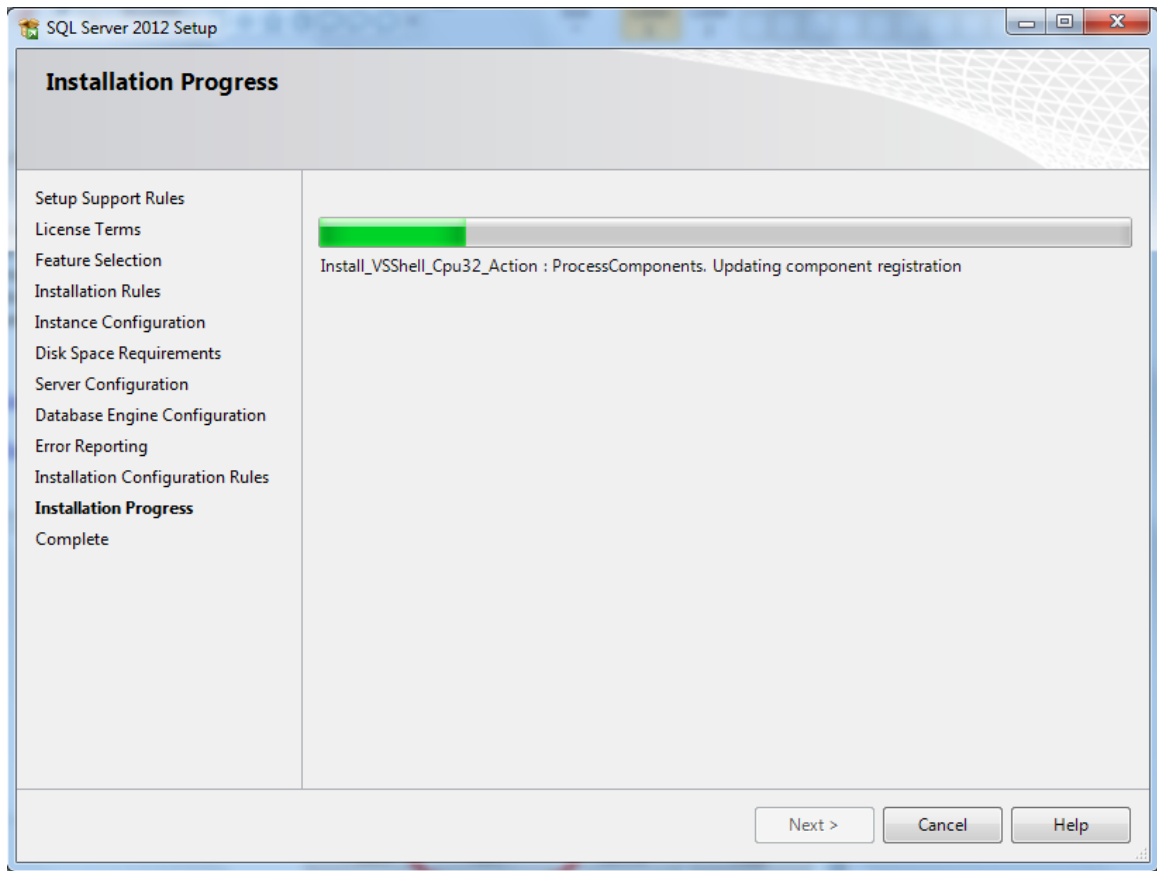
Yonge-PCW7\Yonge (Yonge)

SQL Server administrators have unrestricted access to the Database Engine.

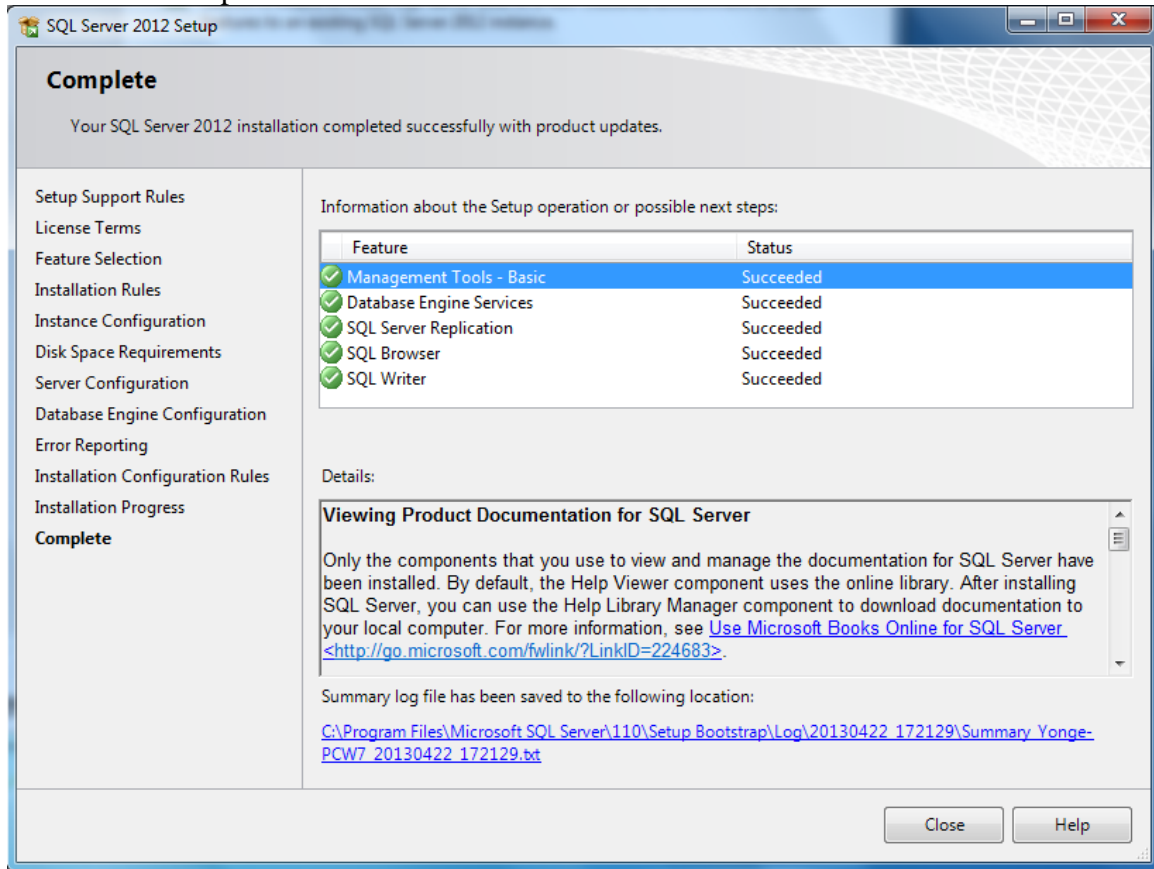
Add Current User Add... Remove

< Back Next > Cancel Help

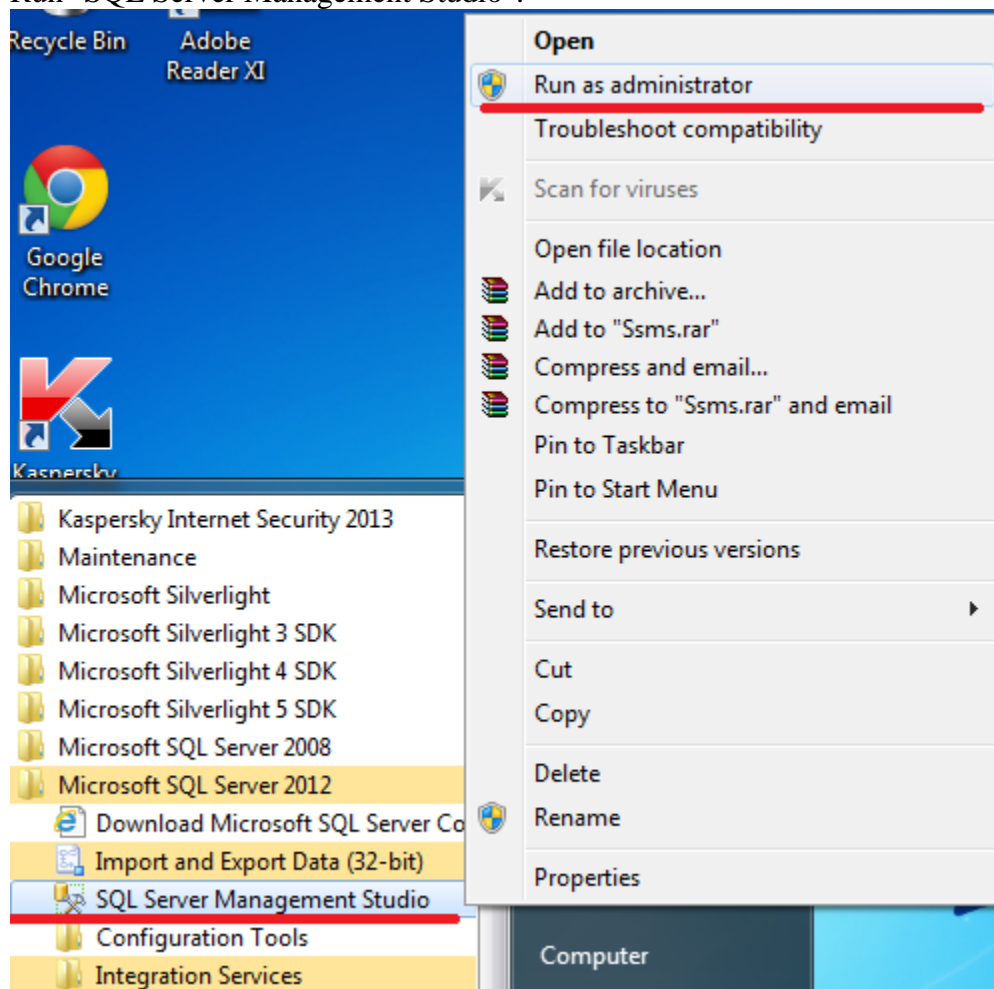




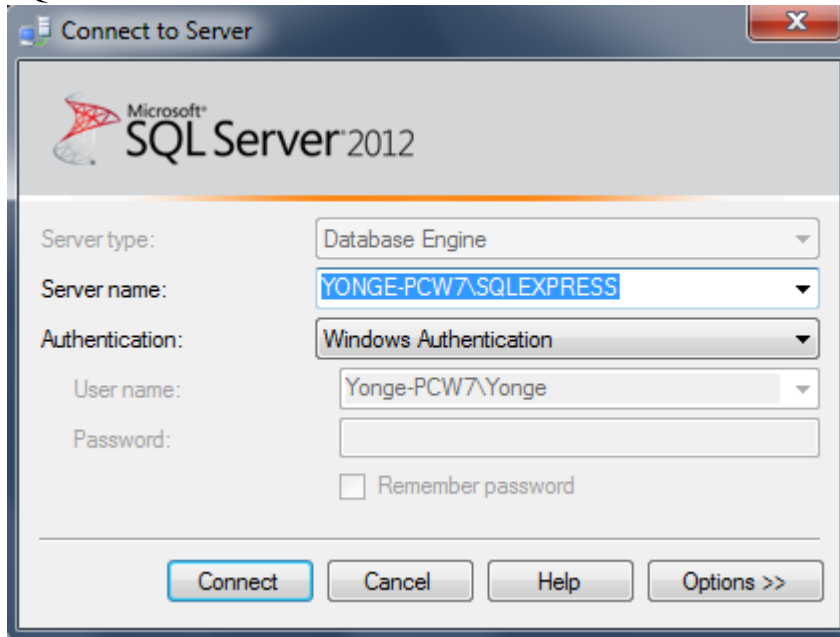
Installation Complete:



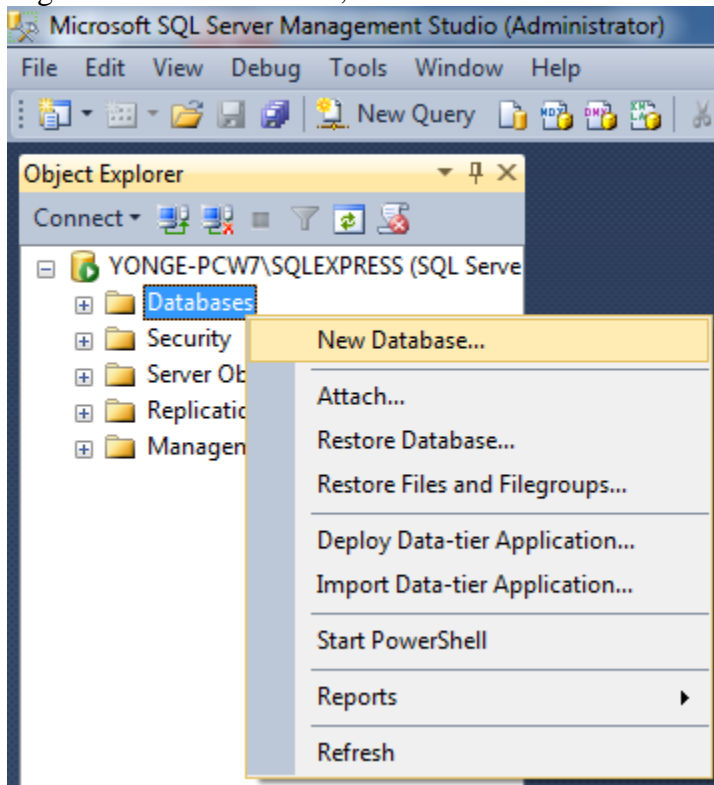
Run "SQL Server Management Studio":



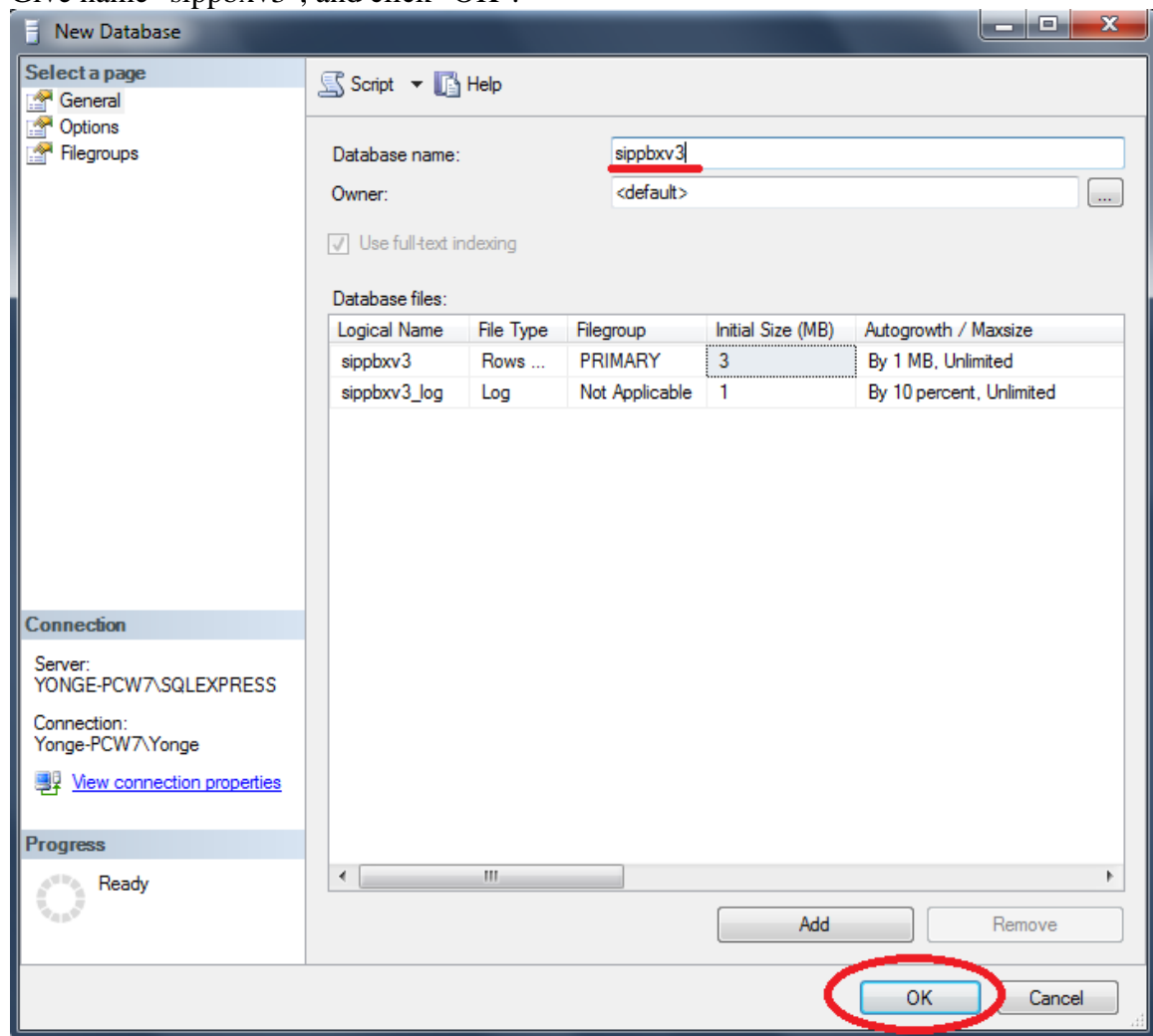
You can use "Windows Authentication" here, and click Next, or use SQL Authentication, then give username sa, password whatever you set when installing SQL server.



Right click on "Database", then choose "New Database":



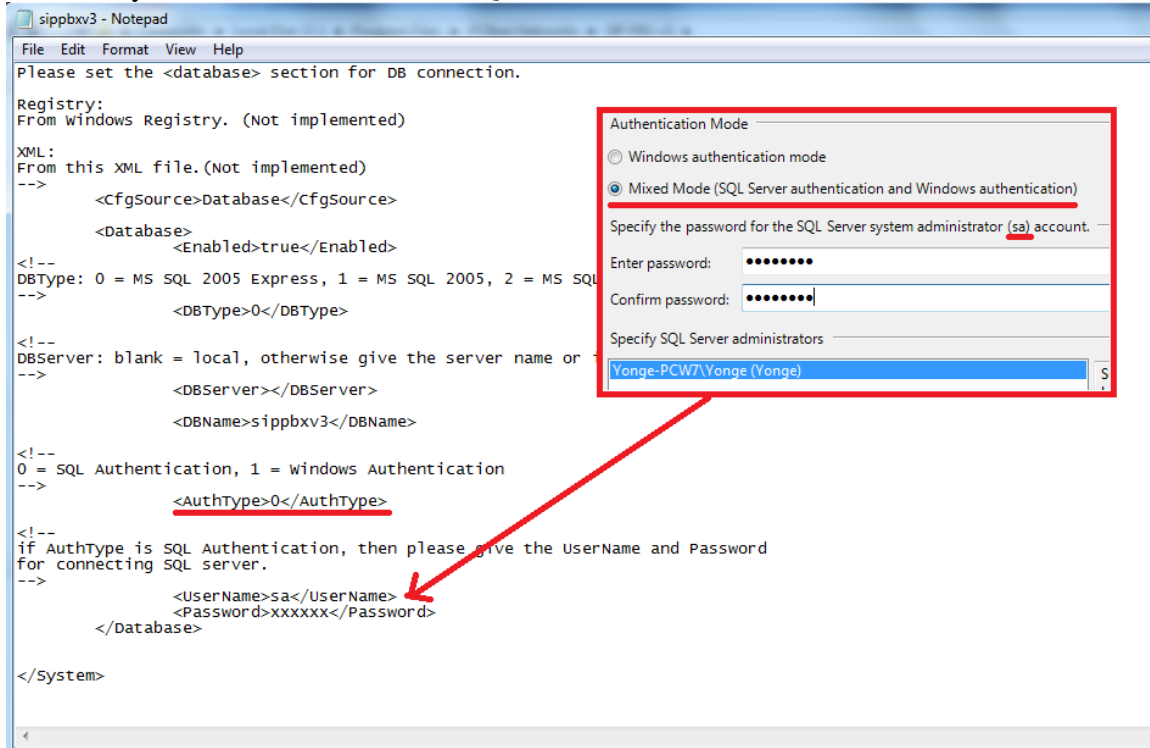
Give name "sippbxv3", and click "OK":



Then database is created, and you can close SQL Server Management Studio.

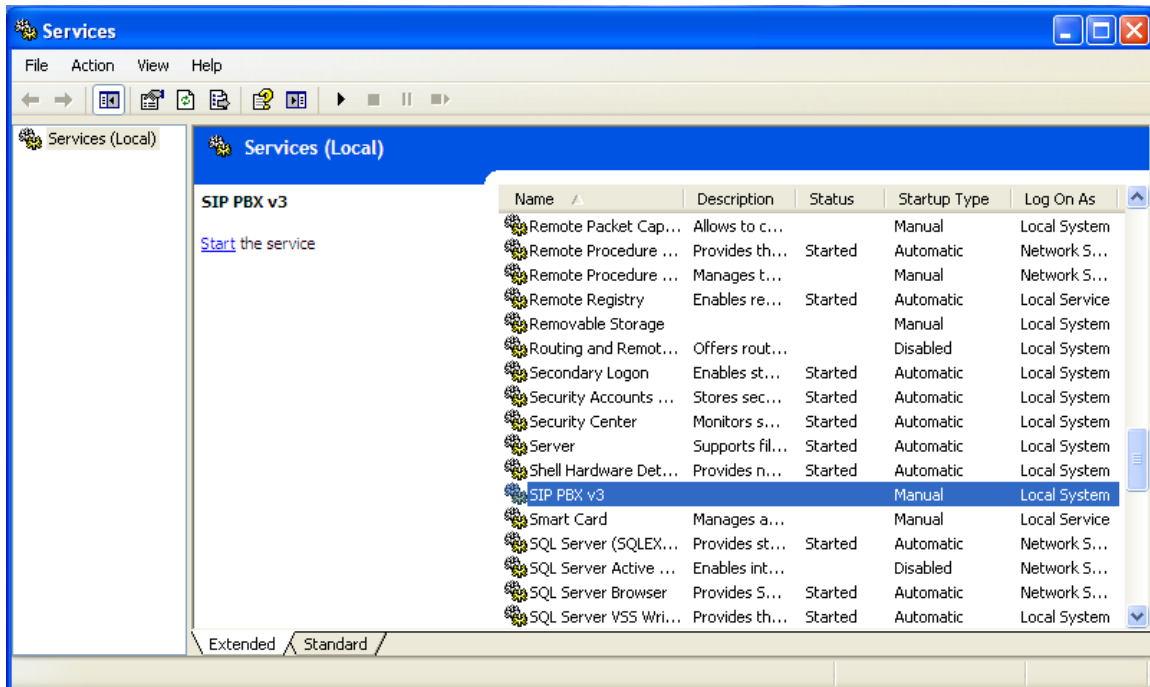
- After the database is created, change the configuration file **sippbxv3.xml** for DB connection. The file can be found in PBX installation folder. Use Notepad or any text editor to open it. Under Windows7 or 2008, in order to change this file, you may need to run Notepad as Administrator first, then open **sippbxv3.xml** in order to save.

In the file, please **set AuthType to 0**, give UserName sa, and password. The password is whatever you set when installation SQL server.

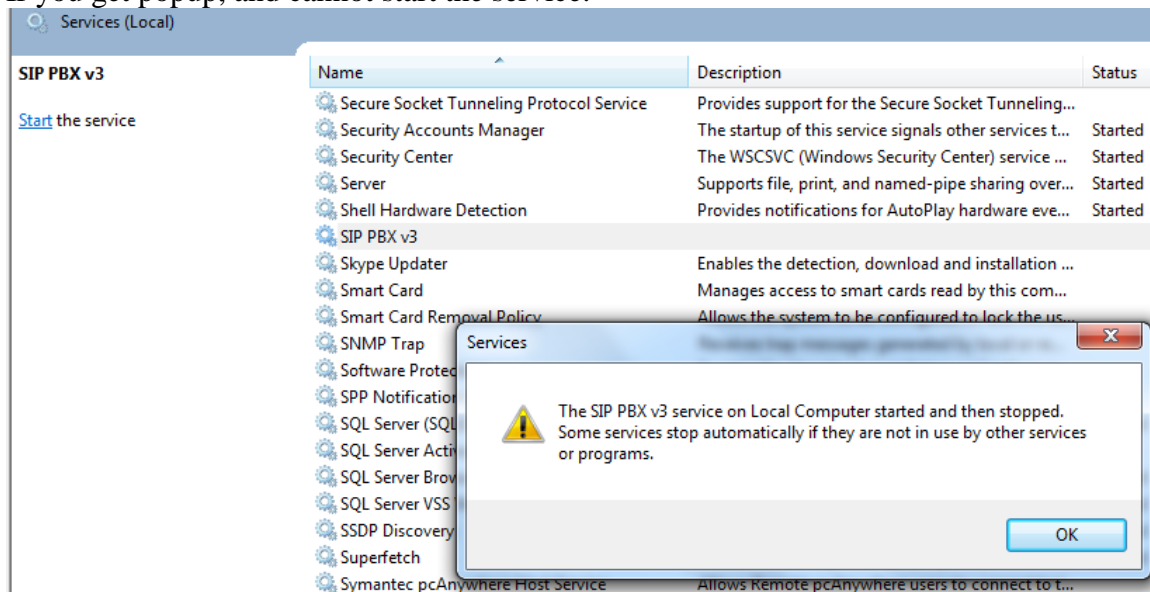


6. Start SIP PBX v3 service

From Control Panel -> Administrator Tool -> Open Windows Services, then find SIP PBX v3 service, then click start(the triangle button):

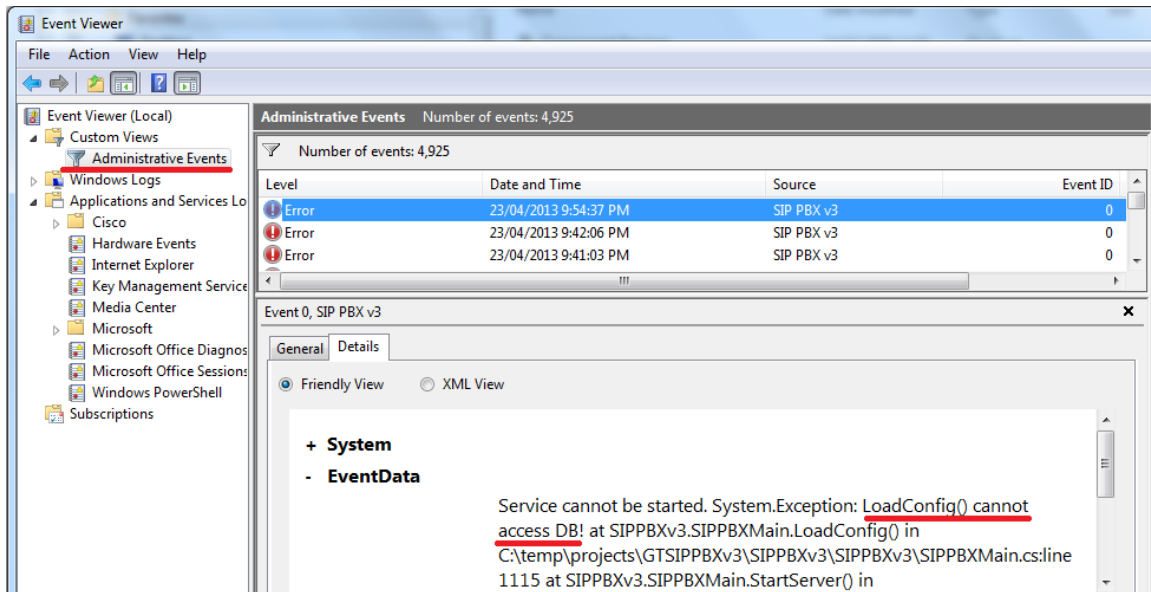


If you get popup, and cannot start the service:

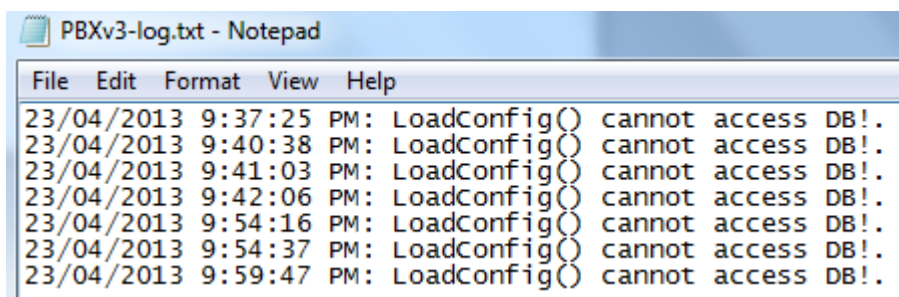
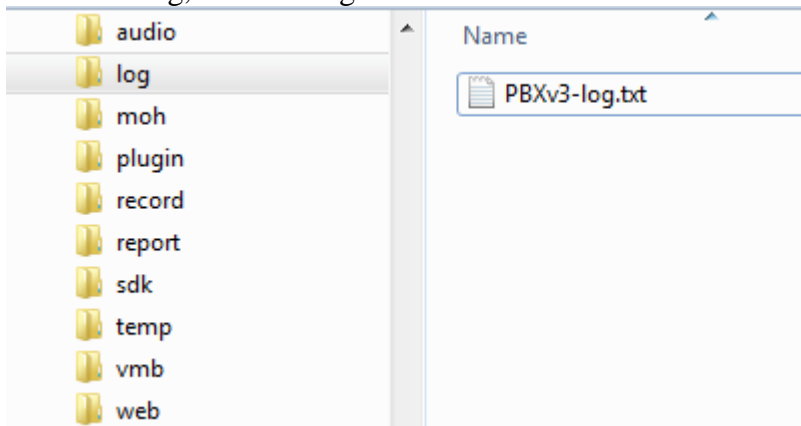


Please check:

a. Event Viewer:

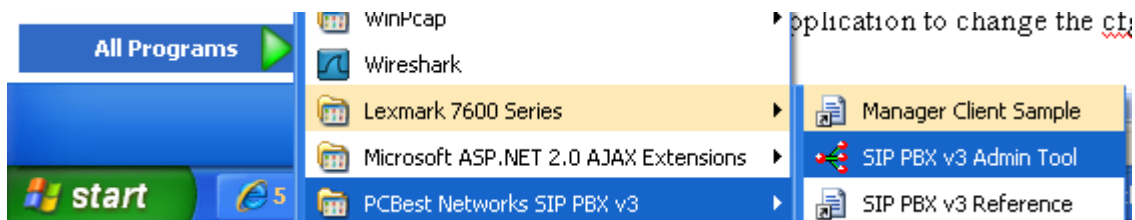


b. PBXv3-log.txt under log folder of PBX installation folder:

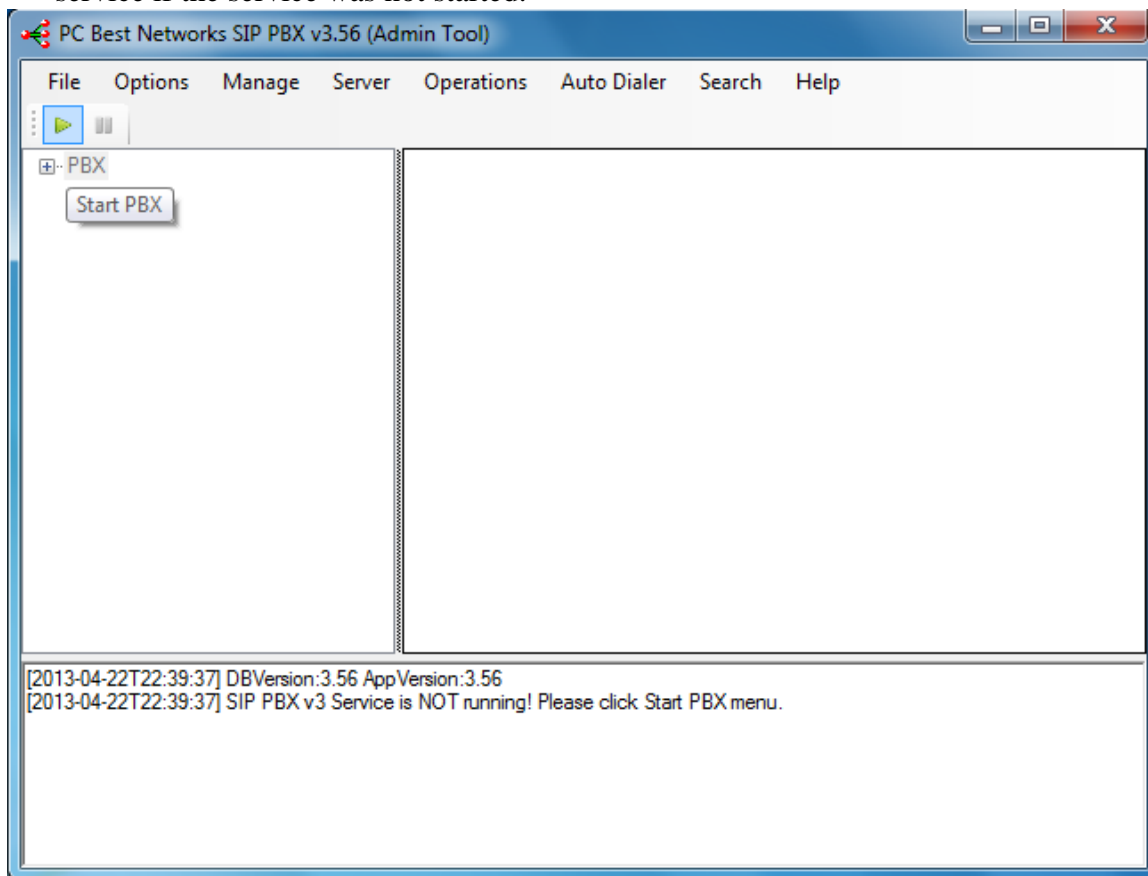


- Run PBX v3 admin tool. **NOTE: if you are using Vista or Windows 7, you need to "Run as administrator" because admin tool needs administrator right to start or stop PBX v3 service.**

PC Best Networks SIP PBX Reference



8. If you see this screen, it means it is working. Click the start button to start the service if the service was not started.



3 PBX Quick Setup Guide

In order to save your time and guide you through the most common scenarios you need to use PCBest SIP PBX for your office environment, this is a quick reference to setup your PBX for Auto Attendant, ACD(Automatical Call Distribution), Outbound Calls, Dial Extension, Virtual Extension, Ring group or Call Parking and etc.

3.1 Common Settings

Before you start, you need to setup the following common settings for all tests.

SIP Accounts(External Lines)

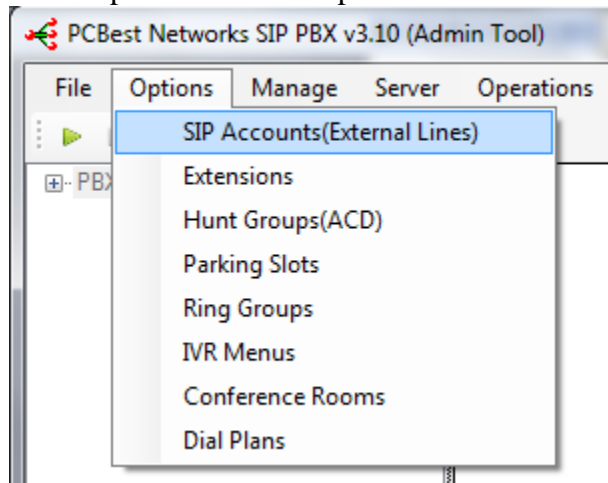
SIP Accounts are the credit info that you can use it to dial out external lines, or receive calls from out lines. For example, you can get a SIP account from ITSP(Internet Telephony Service Provider), then you can make calls to regular phone numbers, or receives calls to your DID.

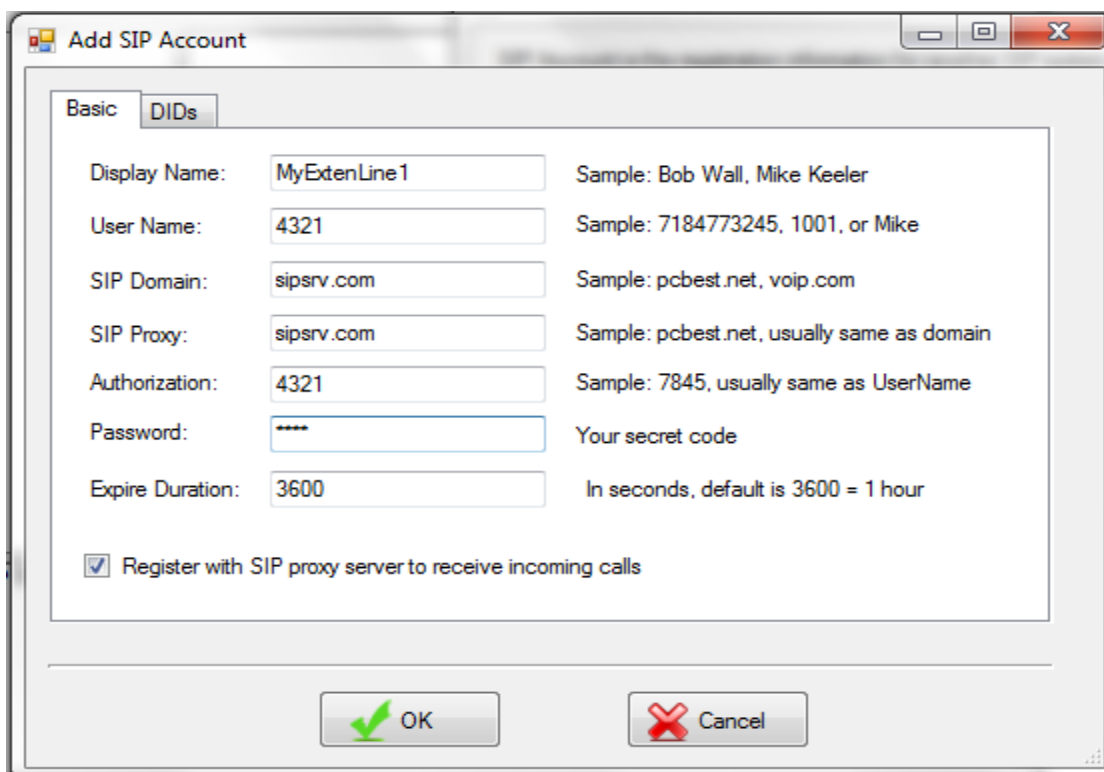
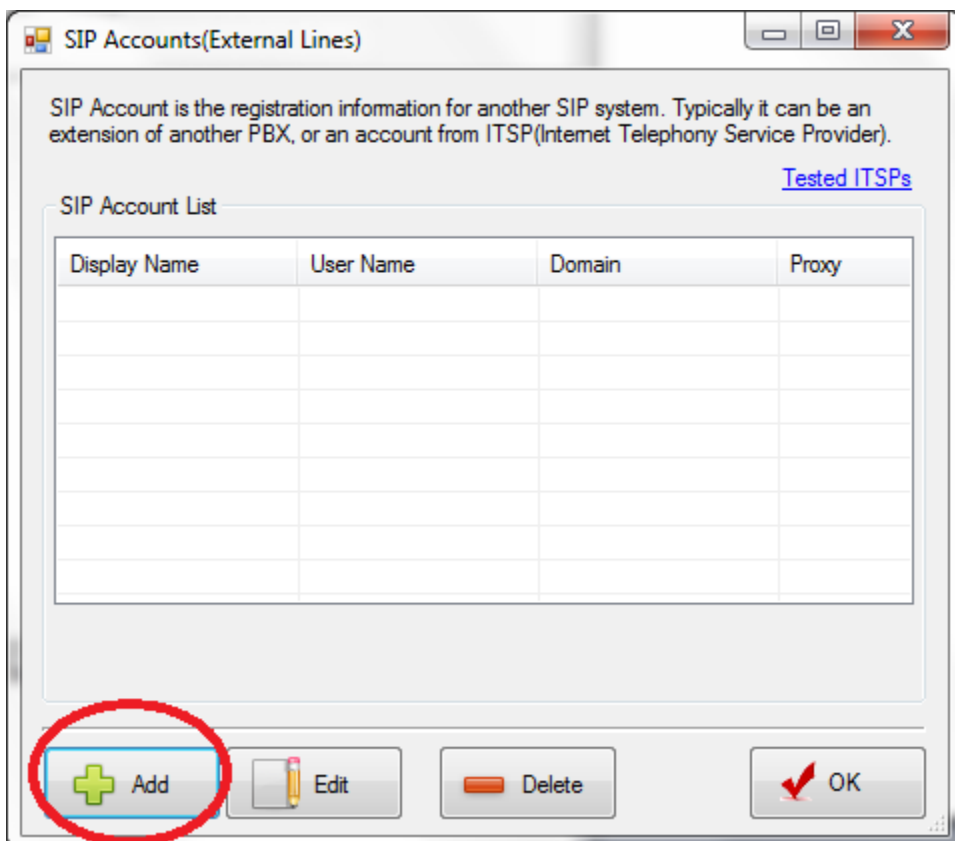
Assume you have a SIP account:

User Name: 4321

Domain: sipsrv.com

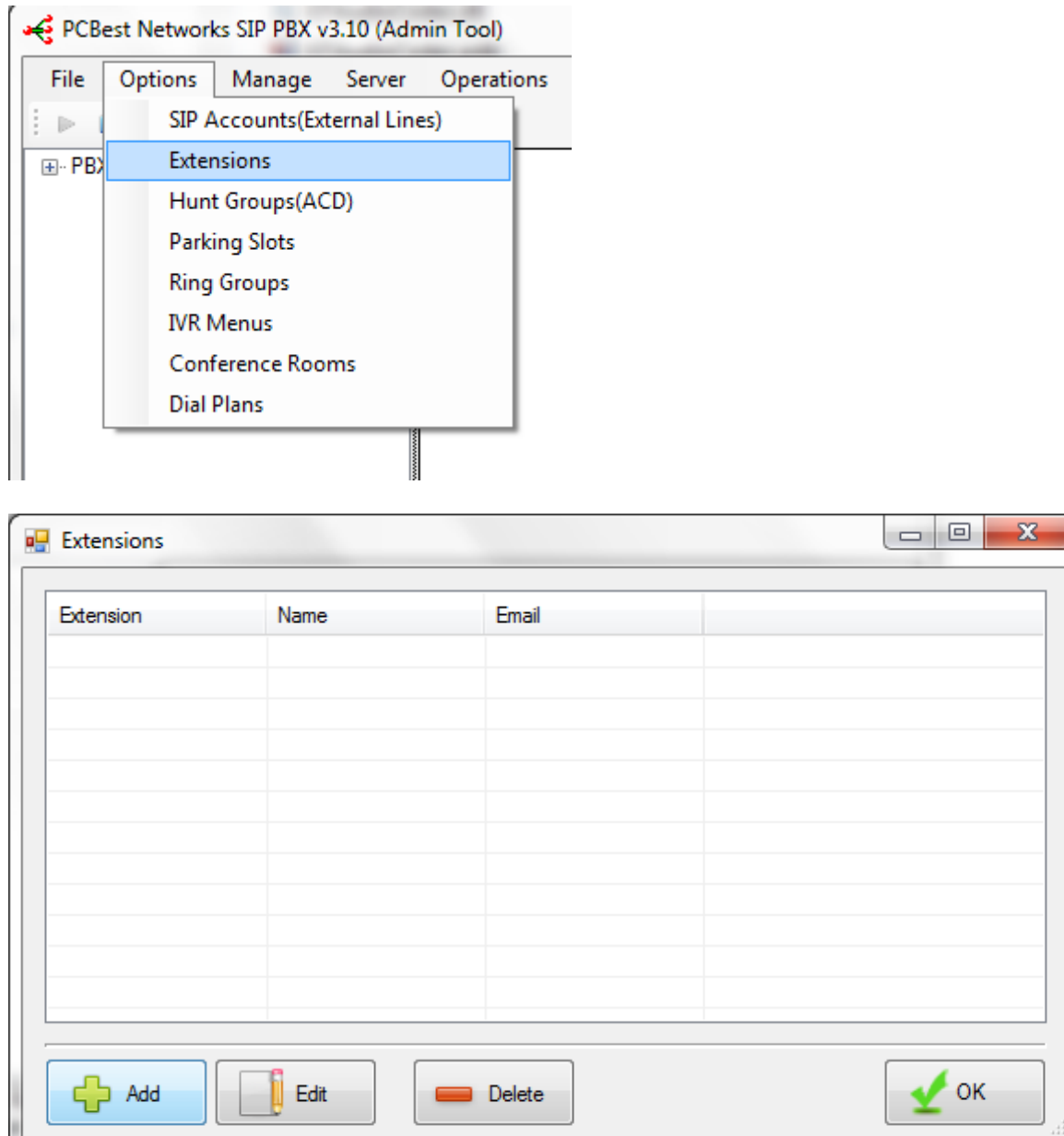
See the pictures to set it up:





Extensions

Extensions are internal phones to handle the calls. Usually extension name are three or four digits length, Like 101, 2010. One extension can also be considered as one SIP account for IP phone, or an outline for another PBX. Assume we setup three extensions here.



Add an extension

Basic | Advanced | Voice Mail Box | Call Forwarding

Extension: (Sample: 101, 1001. Must be unique to the whole PBX, This is also the user name for SIP extension)

User Name: (Sample: Bob wall, Mike Smith)

Password: (The password for SIP extension registration)

Email:

Extension Type:

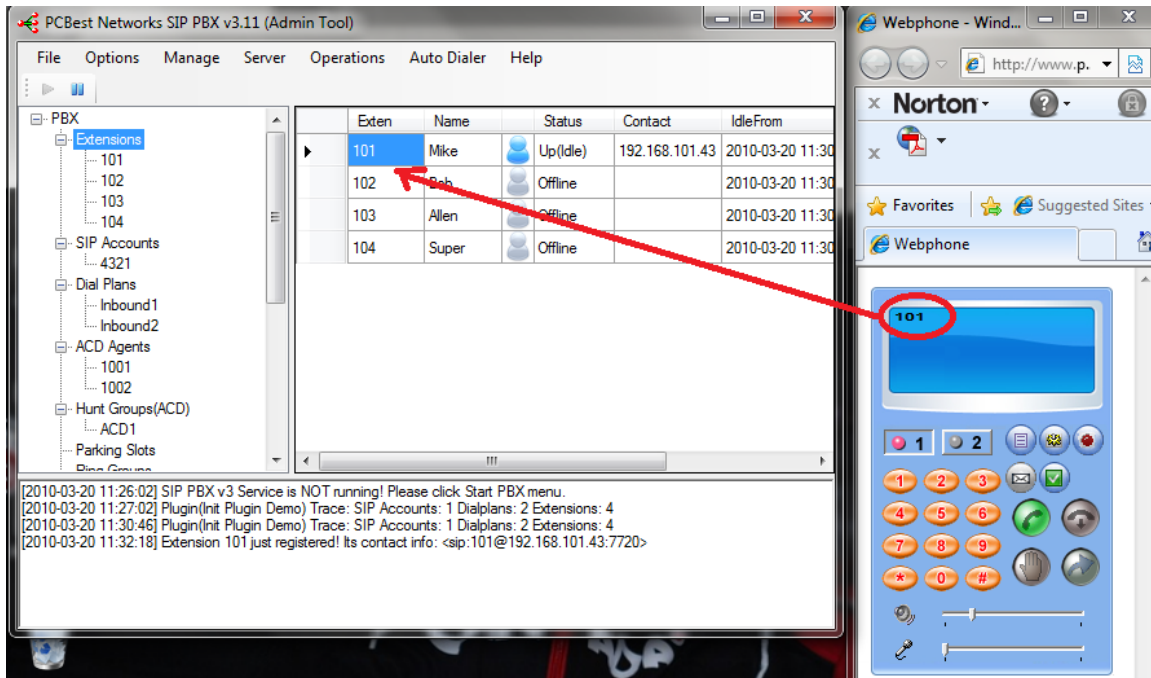
Virtual Extension Outbound Address or Number:

(Use outbound dialplan rule to set outbound number, or use SIP address format like: 123@sipprovider.com)

IP Extension Authorization Type:

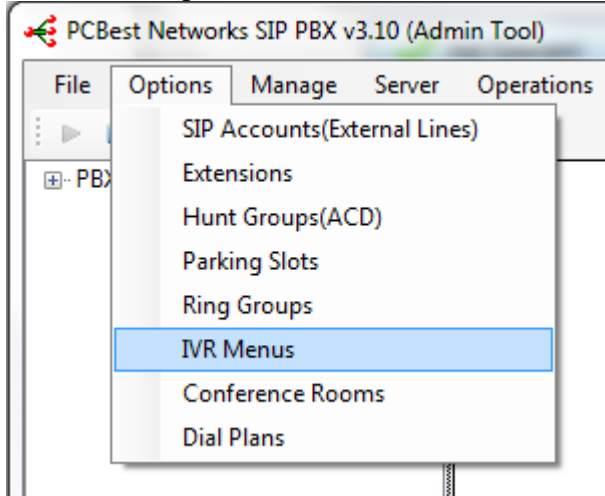
After you have setup three extensions 101, 102, and 103, you need to have 3 ipphones or computers to register on PBX to work as extensions. You can use any SIP hardware phones or softphones, like PCBest SIP ActiveX phone here:

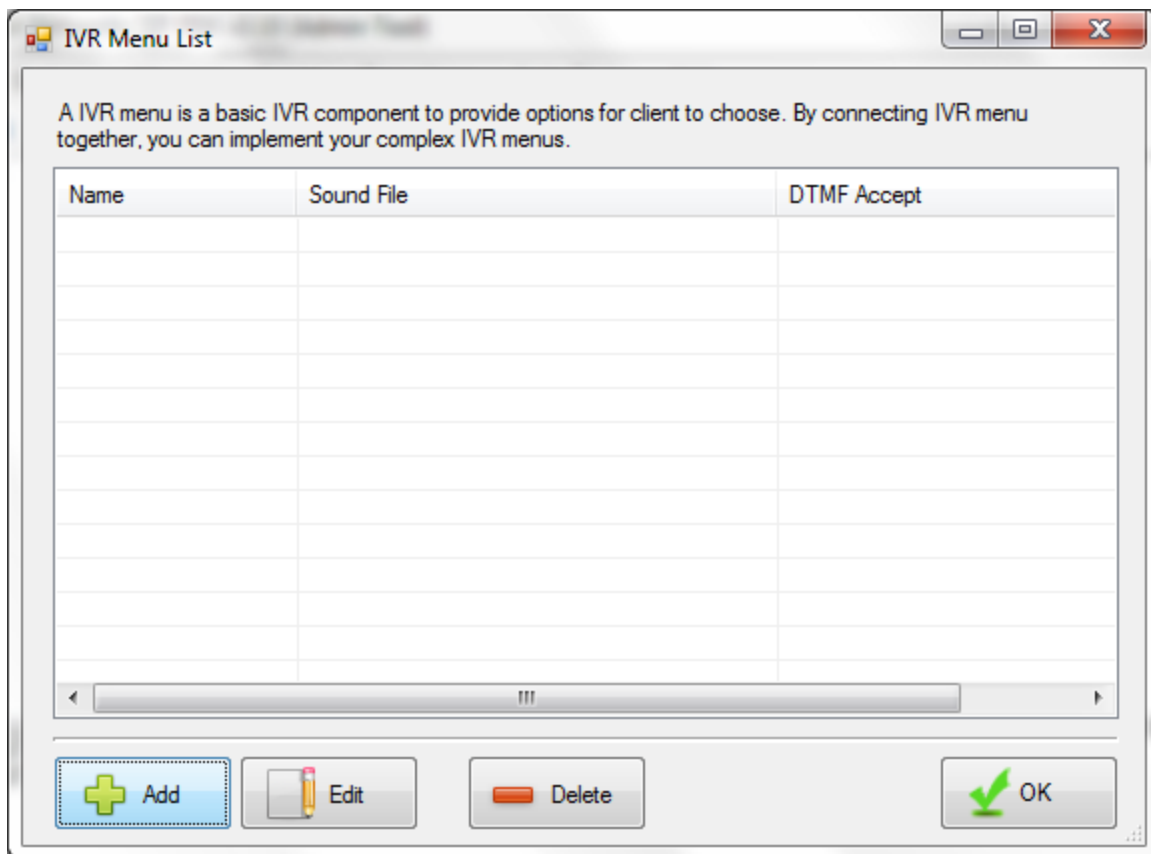
<http://www.pcbest.net/activex.php>



3.2 Auto Attendant

In order to implement Auto-Attendant, we need to set an IVR Menu first to play prompts.





Edit IVR Menu

IVR Menu Name: Please give an unique name, like "IVR1", "MainMenu".

Prompt:

Sample sound: Welcome to ABC company. If you know the extension, please dial it now. Otherwise, press 1 for sales. press 2 for customer support. press 3 for billing department. Stay on the line for operator. Sound file must be 8k 8bit mono mulaw or alaw wave file, or 8k 16bit mono PCM wav file.

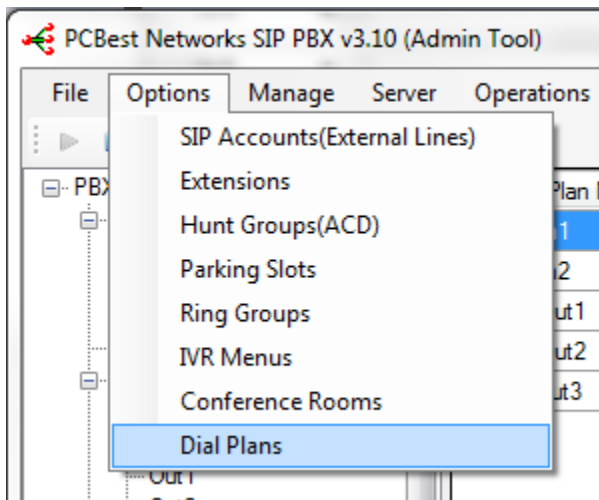
Milliseconds to wait for DTMF keys: Milliseconds.

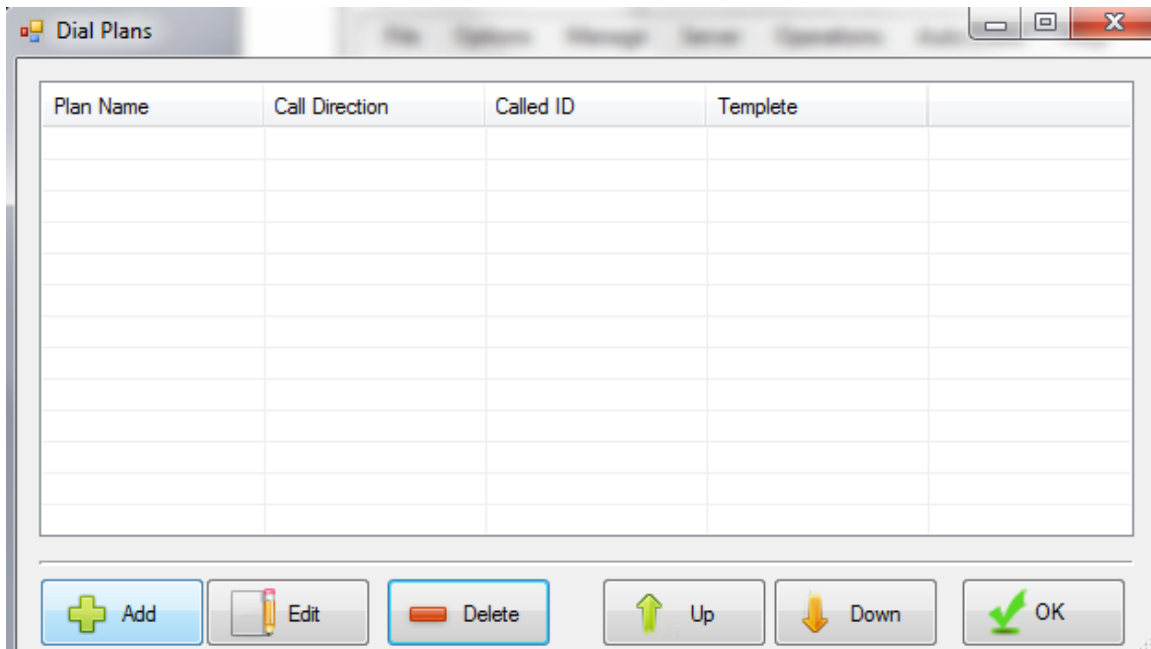
Menu Options

DTMF	Then play:(if blank, jump to action directly)	Action:	Destinations
<input type="text" value="1"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="Dial extension"/>	<input type="text" value="101"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>
<input type="text" value="N/A"/>	<input type="text"/> <input type="button" value="Browse..."/>	<input type="text" value="To another menu"/>	<input type="text"/>

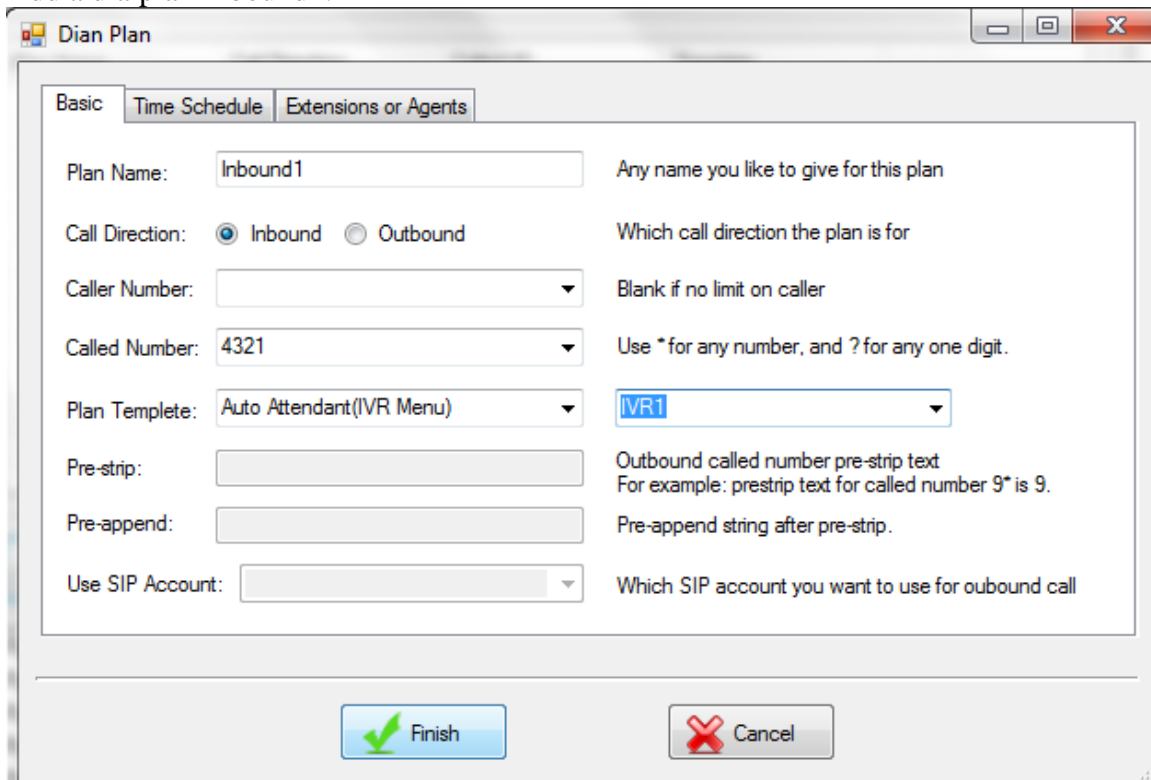
☐ Accept Extensions. Note: if you enable this option, your customer may need to wait Milliseconds on the menu, because of the fact: if you have extensions beginning with 1, like 101,102. Also you set above DTMF menu to accpet 1 to forward calls to ACD group. When users stay on this meanu, and input 1, pbx needs to determine if users want to reach ACD group or want to reach an extension. So pbx will wait above amount of milliseconds to see if users have more inputs. To avoid delay user experience, you can set your extensions beginning with 6 or 7 for example(Leave 8 or 9 for outbound rule).

Then we need to setup an inbound dialplan to connect incoming calls into this IVR menu.





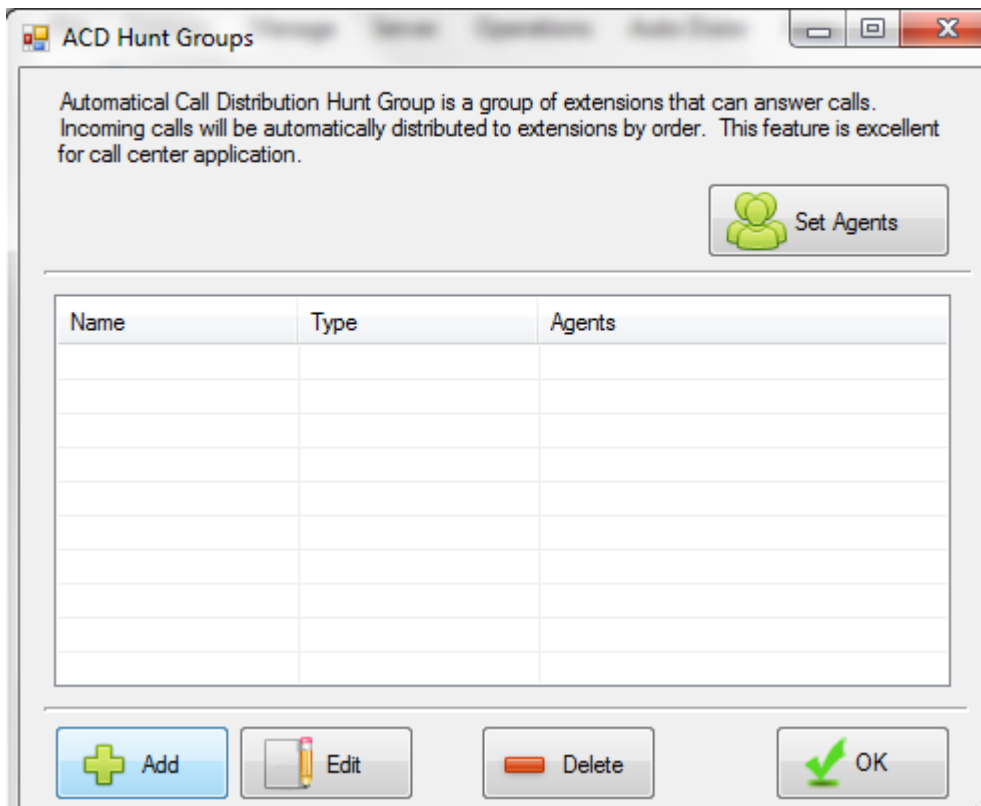
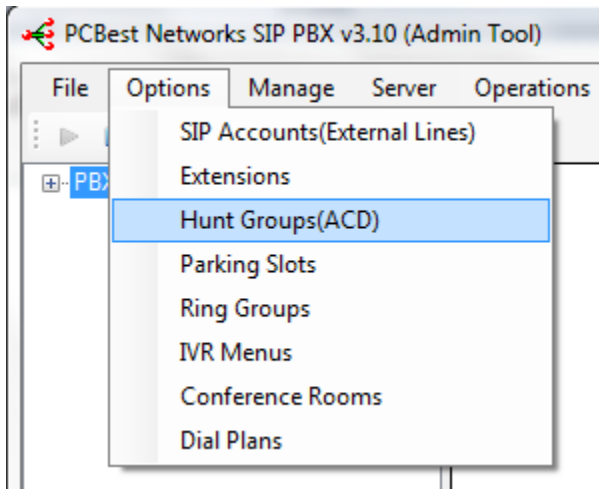
Add a dialplan Inbound1.



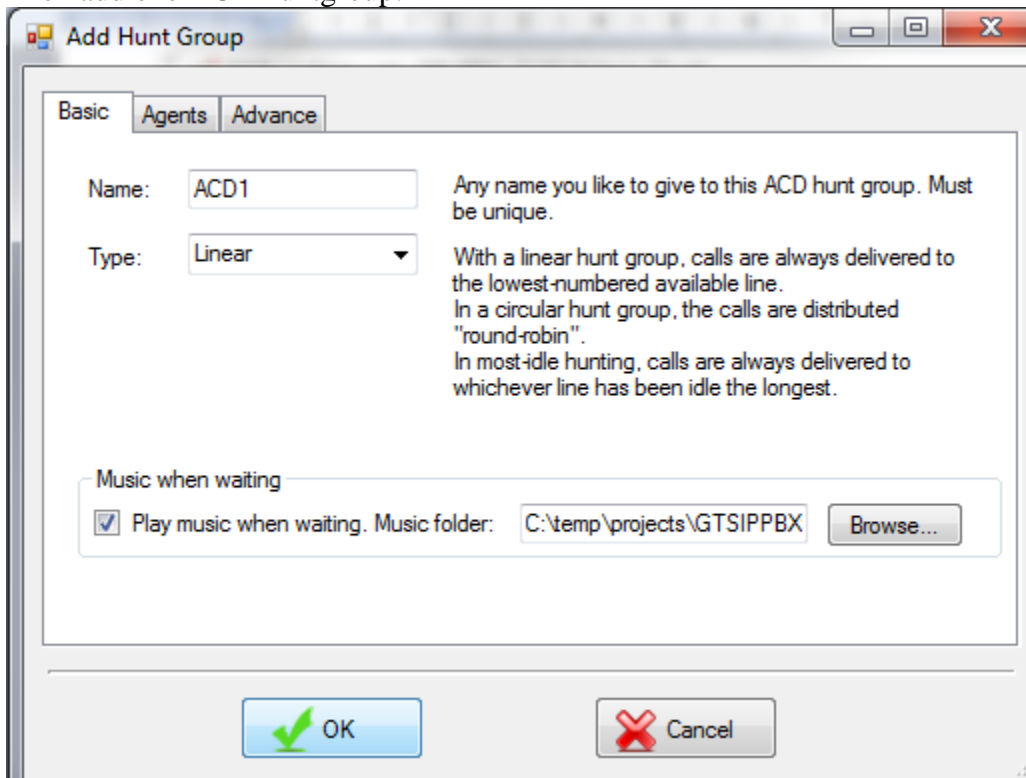
Then when you dial the DID that SIP account 4321 is linked, it will use Dialplan “Inbound1” to handle the call, and call goes to IVR menu “IVR1”.

3.3 ACD(Automatical Call Distribution)

ACD is widely used for call centers. Calls will be automatically queued in ACD group(also called huntgroup), and PBX will try to reach an extension or an agent to answer the call on first in first out order. In order to implement ACD, we need to create an ACD group first.



Then add one ACD huntgroup:

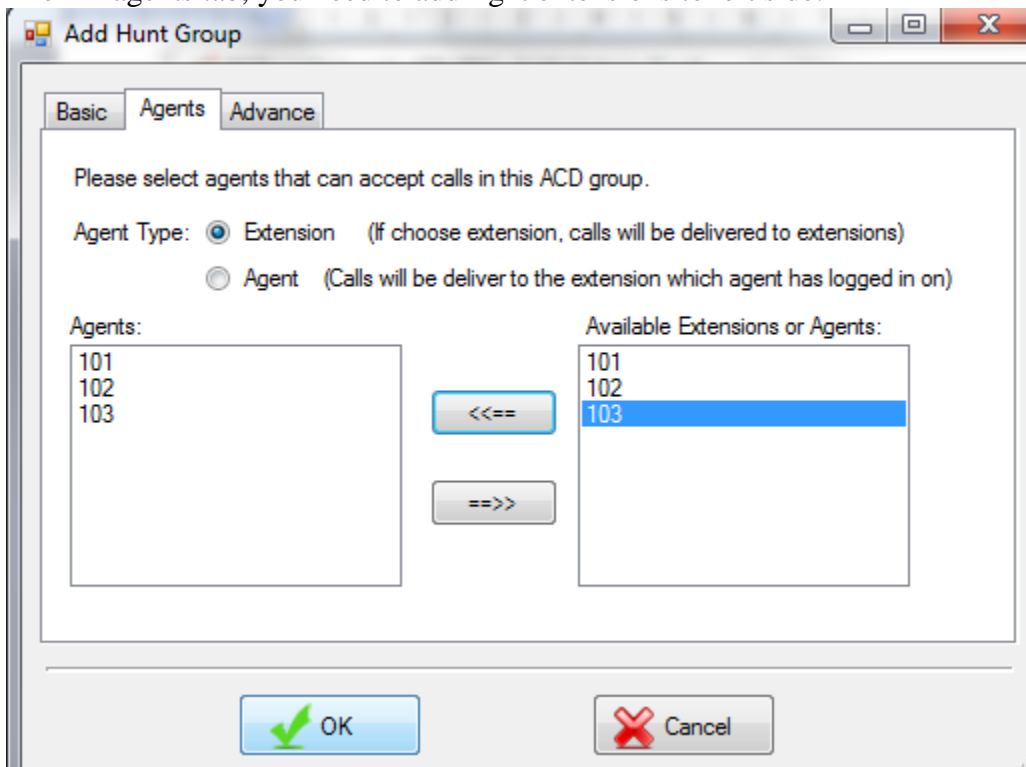


The 'Add Hunt Group' dialog box is shown with the 'Basic' tab selected. It contains the following fields and options:

- Name:** ACD1. A note states: "Any name you like to give to this ACD hunt group. Must be unique."
- Type:** Linear. A dropdown menu with 'Linear' selected. A note explains: "With a linear hunt group, calls are always delivered to the lowest-numbered available line. In a circular hunt group, the calls are distributed 'round-robin'. In most-idle hunting, calls are always delivered to whichever line has been idle the longest."
- Music when waiting:** A section with a checked checkbox 'Play music when waiting' and a text field 'Music folder: C:\temp\projects\GTSIPPBX' with a 'Browse...' button.

At the bottom are 'OK' and 'Cancel' buttons.

Then in agents tab, you need to add right extensions to left side:



The 'Add Hunt Group' dialog box is shown with the 'Agents' tab selected. It contains the following fields and options:

- Please select agents that can accept calls in this ACD group.**
- Agent Type:** Two radio buttons: 'Extension' (selected) with the note "(If choose extension, calls will be delivered to extensions)" and 'Agent' with the note "(Calls will be deliver to the extension which agent has logged in on)".
- Agents:** A list box containing 101, 102, and 103.
- Available Extensions or Agents:** A list box containing 101, 102, and 103, with 103 selected.
- Buttons:** '<<==' and '==>>' buttons between the two list boxes.

At the bottom are 'OK' and 'Cancel' buttons.

Then click OK.

Again, we need to setup an inbound dialplan to connect inbound calls to this ACD huntgroup. Assume we add an inbound dialplan Inbound2 to handle this situation.

The screenshot shows the 'Dian Plan' configuration window with the 'Basic' tab selected. The fields are as follows:

Field	Value	Description/Notes
Plan Name:	Inbound2	Any name you like to give for this plan
Call Direction:	<input checked="" type="radio"/> Inbound <input type="radio"/> Outbound	Which call direction the plan is for
Caller Number:		Blank if no limit on caller
Called Number:	4321	Use * for any number, and ? for any one digit.
Plan Template:	Automatic Call Distribution(ACD)	
Pre-strip:		Outbound called number pre-strip text For example: prestrip text for called number 9* is 9.
Pre-append:		Pre-append string after pre-strip.
Use SIP Account:		Which SIP account you want to use for outbound call

At the bottom, there are 'Finish' and 'Cancel' buttons.

Then any calls goes to 4321 SIP account will be forwarded to ACD1.

3.4 Outbound Calls

Add a dialplan. Give an plan name like OutPlan1. Set it to outbound type.

The screenshot shows a window titled "Dian Plan" with three tabs: "Basic", "Time Schedule", and "Extensions or Agents". The "Basic" tab is active. It contains the following fields and options:

- Plan Name:** OutPlan1 (text input)
- Call Direction:** Inbound (radio), Outbound (radio, selected)
- Caller Number:** (text input)
- Called Number:** 9* (text input)
- Plan Template:** Auto Attendant(IVR Menu) (dropdown menu)
- Pre-strip:** 9 (text input)
- Pre-append:** (text input)
- Use SIP Account:** 4321 4321 (dropdown menu)

At the bottom of the window are two buttons: "OK" (with a green checkmark icon) and "Cancel" (with a red X icon).

Set called number as 9*, and set the SIP account you want to use for dialing out.

Set pre-strip as 9.

It means that any calls go into PBX, which called id starts with 9, the PBX will regard it as an outbound call. PBX will take 9 in the front of called number, and use SIP account 4321 we created to dial out.

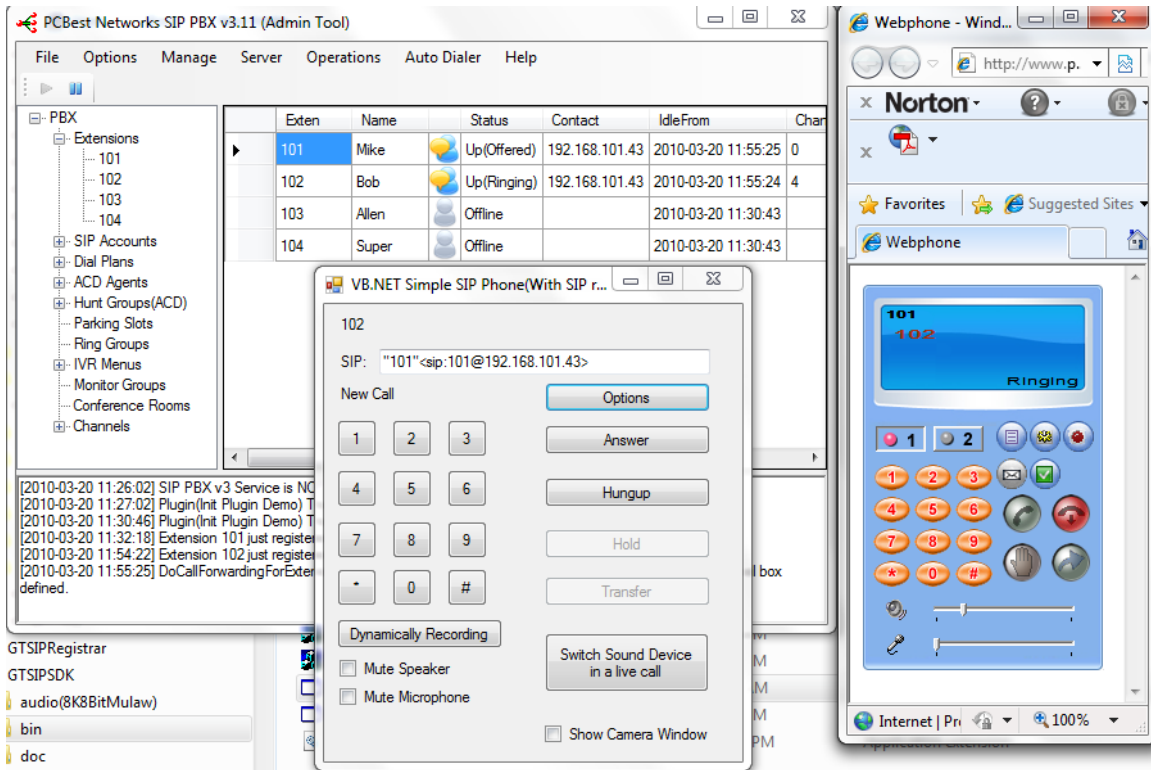
On the sip phone client 101, please dial 9x(x is real phone number you want to reach outside), then PBX should be able to route the call to outside.

3.5 Dial Extension

Extension to extension calls:

You don't need to create any dialplan for extension to extension calls. Assume you have 101 and 102 softphone setup and registered on the PBX. On the softphone 101, you dial 102, then

PC Best Networks SIP PBX Reference



Dial to extension from other options(ACD, IVR menu, ...)

Edit IVR Menu

IVR Menu Name: Please give an unique name, like "IVR1", "MainMenu".

Prompt:

Sample sound: Welcome to ABC company. If you know the extension, please dial it now. Otherwise, press 1 for sales, press 2 for customer support, press 3 for billing department. Stay on the line for operator. Sound file must be 8k 8bit mono mulaw or alaw wave file, or 8k 16bit mono PCM wav file.

Milliseconds to wait for DTMF keys: Milliseconds.

Menu Options

DTMF	Then play:(if blank, jump to action directly)	Action	Destinations
1	<input type="text"/> <input type="button" value="Browse..."/>	Dial extension	101
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1
N/A	<input type="text"/> <input type="button" value="Browse..."/>	To another menu	IVR1

☐ Accept Extensions. Note: if you enable this option, your customer may need to wait Milliseconds on the menu, because of the fact: if you have extensions beginning with 1, like 101,102. Also you set above DTMF menu to accpet 1 to forward calls to ACD group. When users stay on this meanu, and input 1, pbx needs to determine if users want to reach ACD group or want to reach an extension. So pbx will wait above amount of milliseconds to see if users have more inputs. To avoid delay user experience, you can set your extensions beginning with 6 or 7 for example(Leave 8 or 9 for outbound rule).

3.6 Virtual Extension

Virtual extension is a kind of extension which pointed to an outside phone number. Let us create an extension which has virtual extension type.

Edit Extension

Basic | Advanced | Voice Mail Box | Call Forwarding

Extension: 104 (Sample: 101, 1001. Must be unique to the whole PBX, This is also the user name for SIP extension)

User Name: Super (Sample: Bob wall, Mike Smith)

Password: **** (The password for SIP extension registration)

Email:

Extension Type: Virtual

Virtual Extension Outbound Address or Number: 91234567
(Use outbound dialplan rule to set outbound number, or use SIP address format like: 123@sipprovider.com)

IP Extension Authorization Type: Proxy

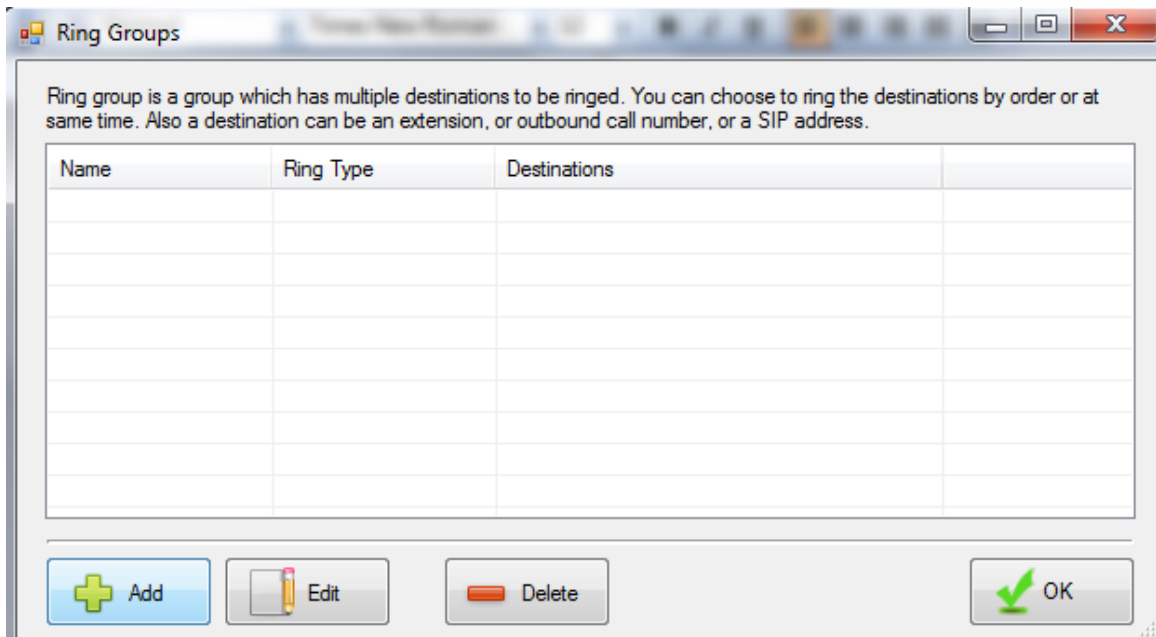
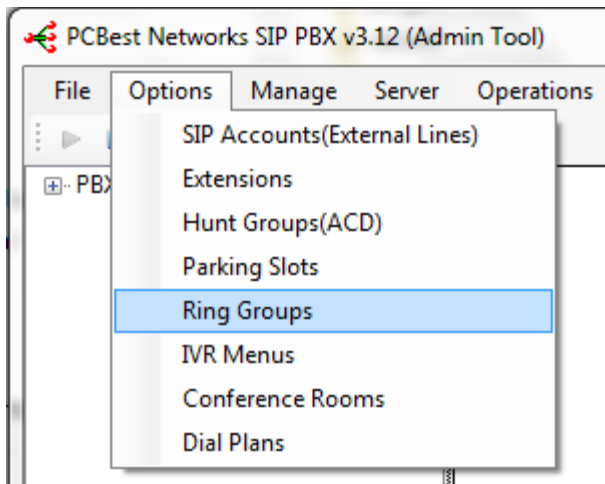
Update Extension Cancel

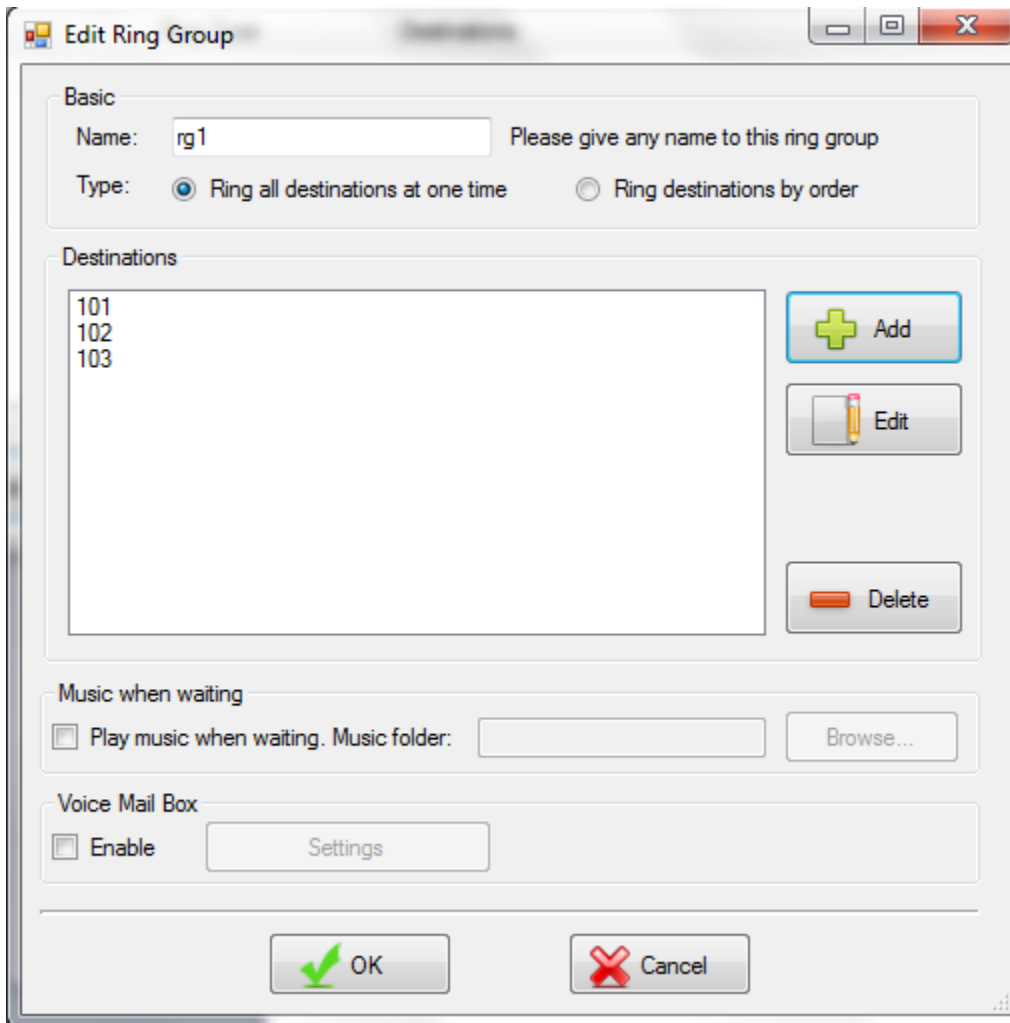
We set 91234567 here, which means using outbound plan 9*. When calls go to this extension, PBX will try to reach outside number 1234567.

3.7 Ring group

Ring group is a group of extensions or agents that can be ringed(called) by order or same time. Ring group doesn't work like ACD. ACD holds calls until extensions or agents are available to answer the call. Ring group doesn't really hold the calls for long time. It will try to ring the destinations, and the first destination which answered call will be connected to the caller.

Set up a ring group first. Assume its name is rg1.

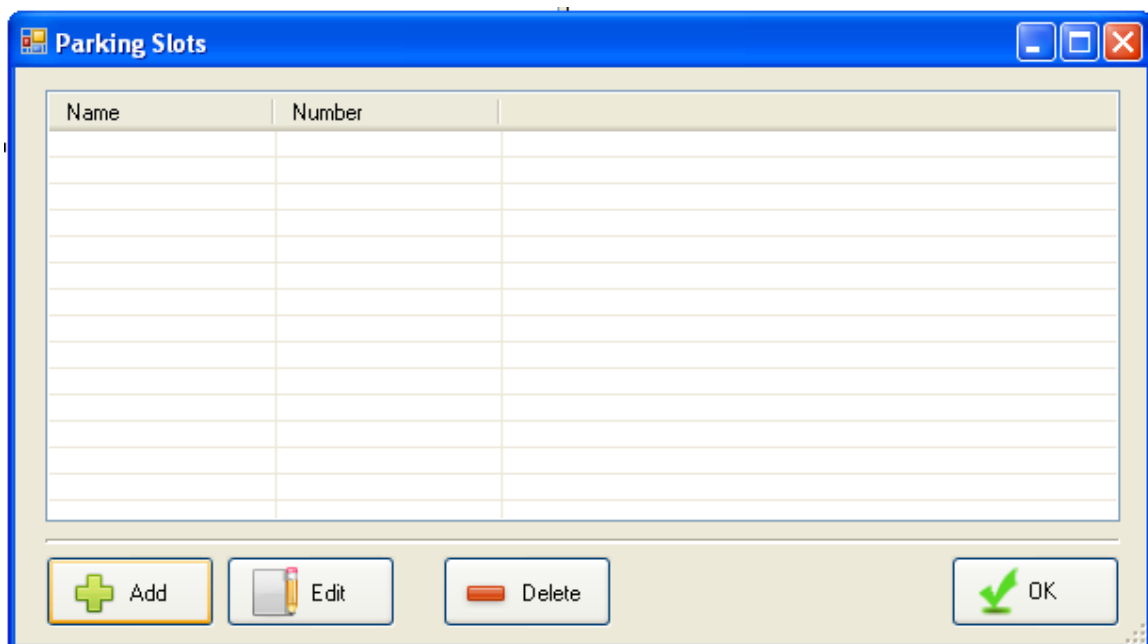
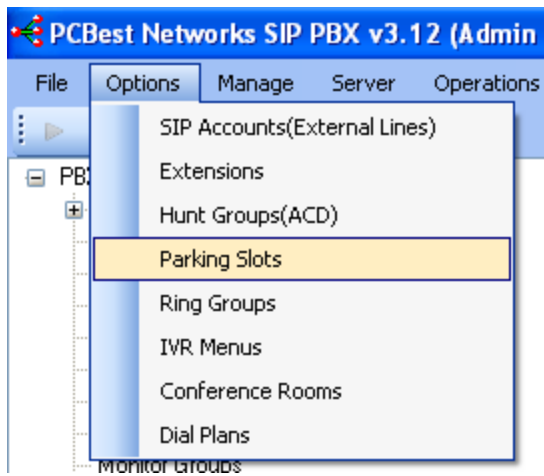




Three extensions 101, 102, 103 are added into ring group rg1. Then we can set up an inbound dialplan, to forward calls to this ring group. When a call comes in and reach this ring group, pbx will ring extensions 101, 102, 103 at same time.

3.8 Call Parking

Call Parking is used to park a call. You must define a call parking slot first to allow the call to park, then later the call can be picked up by another extension or agent.



Edit Parking Slot

Parking slot is used to park a call, which can be picked up later by dialing the parking slot's number.

After an agent answered a call, he/she can input the parking slot's number to park this call. Once the call is parked successfully, the agent's call will be automatically disconnected, and another agent can dial the parking slot's number to pick up that call.

Basic **Advance**

Parking Slot Name: Any name. Sample: Slot 1

Number: Sample: *61, #10,...

Music On Hold

☐ Play music when call parked

Music files from:

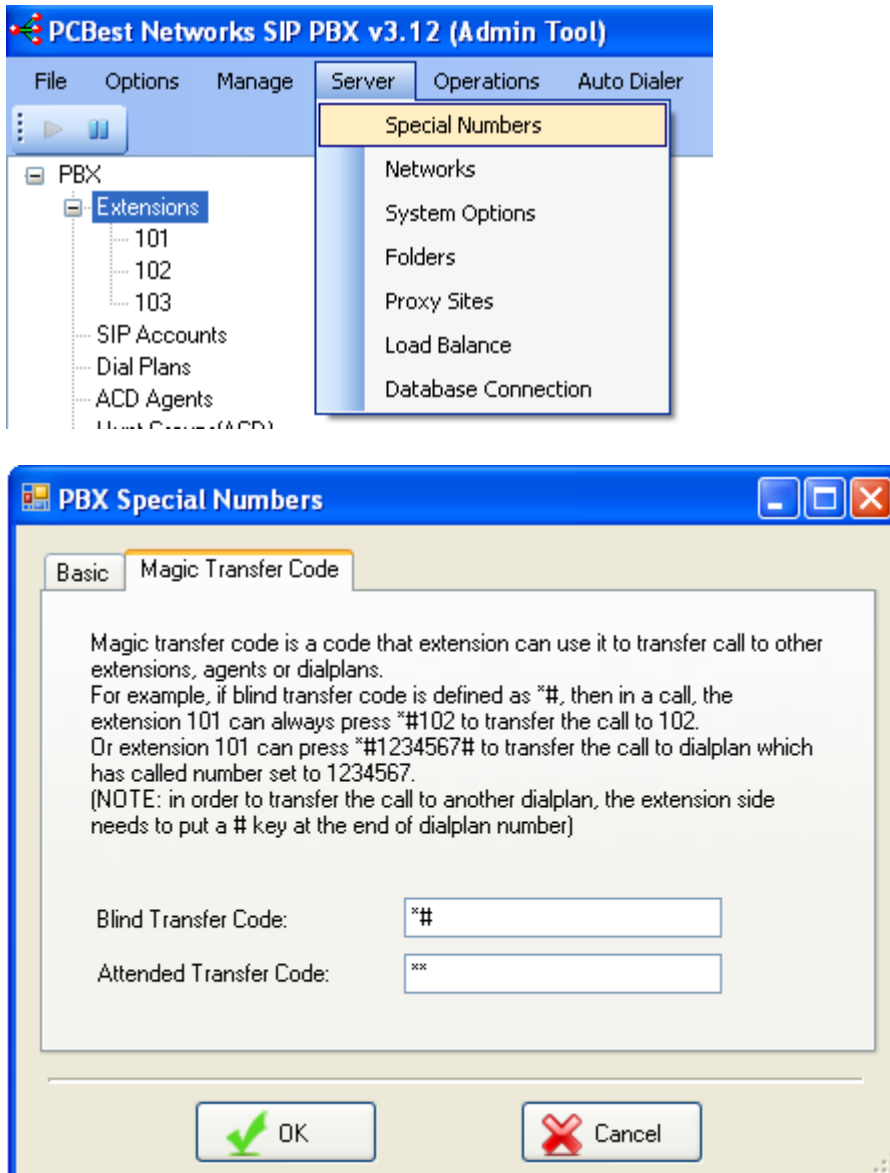
After defined a Parking Slot “PK1”, you can try an incoming call which is transferred into an extension or agent. When extension pressed *61, the call should be parked. Another extension should be able to pick up this call by dialing *61 into PBX.

3.9 Magic Transferring Code (ONLY V3)

Magic Transferring Code is used by extensions to transfer current calls to another extension. There are two kinds of transferring:

1. Blind Transfer
2. Attended Transfer

You don't need to define anything. Magic transferring code default works. Blind transfer code is defined as *#, and Attended transfer code is defined as **.



3.10 FXO/FXS or Digital Gateway

PCBest SIP PBX works with most standard FXO/FXS or Digital Gateways. You can configure gateway works as a peer of PCBest SIP PBX.

Assume gateway works at 192.168.1.10, and PCBest SIP PBX runs at 192.168.1.20. On the gateway, you need to forward the incoming calls into IP address 192.168.1.20, and on the PCBest SIP PBX, you need to set up a fake SIP account that points to gateway's IP address:

The screenshot shows the 'Add SIP Account' dialog box with the 'Basic' tab selected. The fields are as follows:

Field	Value	Sample
Display Name:	ToGW	Bob Wall, Mike Keeler
User Name:	100	7184773245, 1001, or Mike
SIP Domain:	192.168.1.10	pcbest.net, voip.com
SIP Proxy:	192.168.1.10	pcbest.net, usually same as domain
Authorization:	100	7845, usually same as UserName
Password:	xxxx	Your secret code
Expire Duration:	3600	In seconds, default is 3600 = 1 hour

At the bottom, there is a checkbox labeled 'Register with SIP proxy server to receive incoming calls'. This checkbox is circled in red. Below it, red text reads: 'Any as GW doesn't check your authentication' and 'uncheck this option because it is a fake account'. The dialog box has 'OK' and 'Cancel' buttons at the bottom.

By doing this, you setup a peer which is connected to your gateway. Next step, you need to setup an outbound dialplan to use this sip account to forward extension calls into gateway.

Dian Plan

Basic | Time Schedule | Extensions or Agents

Plan Name: Any name you like to give for this plan

Call Direction: ☐ Inbound ☒ Outbound Which call direction the plan is for

Caller Number: Blank if no limit on caller

Called Number: Use * for any number, and ? for any one digit.

Plan Template:

Pre-strip: Outbound called number pre-strip text
For example: prestrip text for called number 9* is 9.

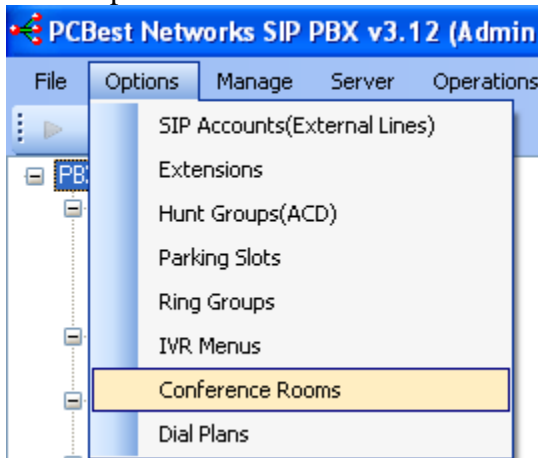
Pre-append: Pre-append string after pre-strip.

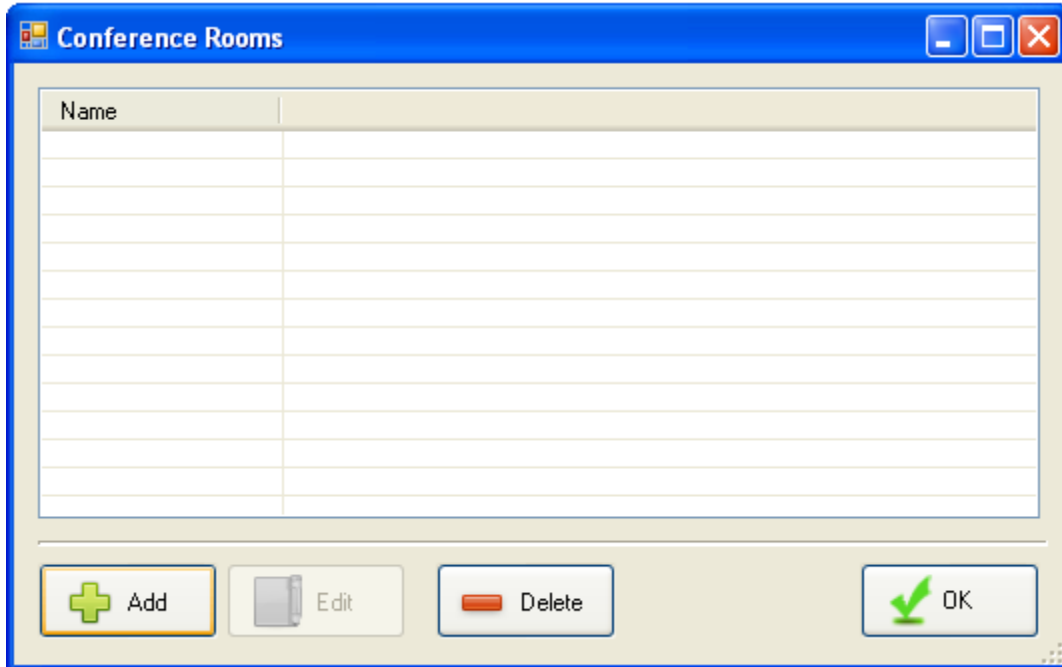
Use SIP Account: Which SIP account you want to use for outbound call

OK Cancel

3.11 Conference Room

You can define a conference room, then forward multiple calls into one conference room, so multiple ends can have a conference call.





Then you can define a dialplan to forward incoming calls into this conference room.

3.12 Inbound 2 Outbound

Sometimes you need to convert an inbound call to outbound call directly.

Because only extensions can call outbound dialplan, so you can achieve this by two ways:

1. Create a virtual extension. In the virtual extension destination address, you can input *, means directly inbound call(dialplan)'s called id to find out proper dialplan. You can give *@outbound-dialplan-name to specify using which dialplan. You can also give sip

address like <sip:*@sipaccount-domain> to route call out by specific sip account. More, giving a sip ip address like <sip:*@ip-address> should work too.

Add an extension

Basic | Advanced | Voice Mail Box | Call Forwarding

Extension: (Sample: 101, 1001. Must be unique to the whole PBX. This is also the user name for SIP extension)

User Name: (Sample: Bob wall, Mike Smith)

Password: (The password for SIP extension registration)

Email:

Extension Type:

Virtual Extension Outbound Address or Number:
 (Use outbound dialplan rule to set outbound number, sample like 9123456, if you have defined outbound dialplan for 9*. Or use SIP address format like: 123@sipprovider.com, or *@sipprovider.com. * means forward the original called id. You can also use *@outbound-dialplan-name, which means forwarded original called id to an outbound dialplan)

IP Extension Authorization Type:

2. Use call forward inbound dialplan

Create an inbound dialplan, set call template to call forward, then choose an outbound dialplan for call forwarding.

Note, for this call forwarding inbound dialplan, please adjust its order in the dialplan list, and make it up and be front of outbound dialplan.

Dial Plan

Basic | Time Schedule | Extensions or Agents

Plan Name: Any name you like to give for this plan

Call Direction: ☒ Inbound ☐ Outbound Which call direction the plan is for

Caller Number: Blank if no limit on caller

Called Number: 9 Use * for any number, and ? for any one digit.

Plan Template: Call Forward OutPlan1

Pre-strip: Outbound called number pre-strip text
For example: prestrip text for called number 9* is 9.

Pre-append: Pre-append string after pre-strip.

Use SIP Account: Which SIP account you want to use for outbound call

3.13 Setup a music server

Create an inbound dialplan, and choose call plan template to "Music Server", then give the name of music file folder.

Dial Plan

Basic | Time Schedule | Extensions or Agents

Plan Name: Any name you like to give for this plan

Call Direction: ☒ Inbound ☐ Outbound Which call direction the plan is for

Caller Number: Blank if no limit on caller

Called Number: 8765 Use * for any number, and ? for any one digit.

Plan Template: Music Server C:\temp\projects\GTSIPPBXv3\lr

Pre-strip: Outbound called number pre-strip text
For example: prestrip text for called number 9* is 9.

Pre-append: Pre-append string after pre-strip.

Use SIP Account: Which SIP account you want to use for outbound call

Finish Cancel

3.14 Echo Test for IP extension

Create an inbound dialplan, and choose call plan template to "Echo Test".
 IP extensions can call this inbound dialplan to see if voice can be returned back in time.
 Sometimes we use this feature to detect network problem like one-way audio.

Dian Plan

Basic | Time Schedule | Extensions or Agents

Plan Name: Any name you like to give for this plan

Call Direction: ☒ Inbound ☐ Outbound Which call direction the plan is for

Caller Number: Blank if no limit on caller

Called Number: Use * for any number, and ? for any one digit.

Plan Template:

Pre-strip: Outbound called number pre-strip text
For example: prestrip text for called number 9* is 9.

Pre-append: Pre-append string after pre-strip.

Use SIP Account: Which SIP account you want to use for outbound call

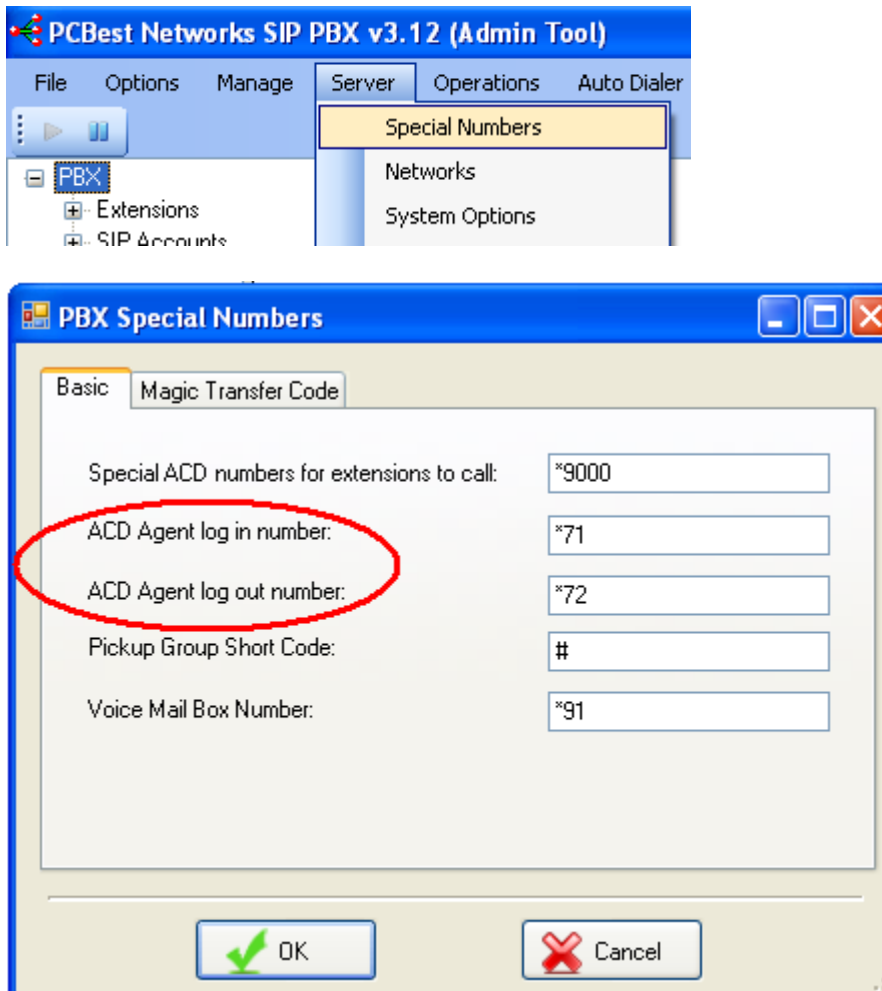
4 PBX Advanced Call Center Features

PCBest SIP PBX can be used as a call center environment. As described in 3.3, Automatic Call Distribution group can allow you to set up a group of agents to answer incoming calls.

4.1 Setting up ACD agents

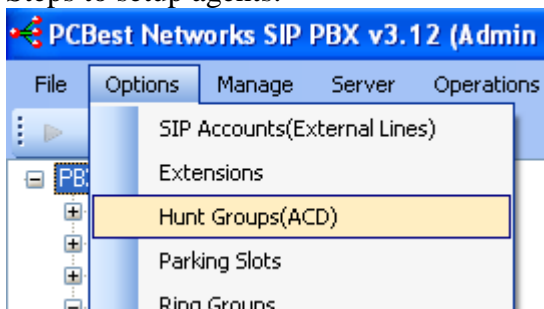
What is an agent? An agent is **NOT** an extension.

An extension is a physical phone, but an agent is a real person to work on an extension. So there may be more than one agent working on the same extension. Usually in a call centre environment, an agent will start to work by login at one of the extension. PBX defines special phone numbers for agents to login and logout at extensions.




Agents can call above special login and logout numbers from any extension to indicate they are at that extension or not.

Steps to setup agents:







ACD Hunt Groups

Automatic Call Distribution Hunt Group is a group of extensions that can answer calls. Incoming calls will be automatically distributed to extensions by order. This feature is excellent for call center application.


 Set Agents

Name	Type	Agents





 Add  Edit  Delete  OK

ACD Agents

ACD agents are the people who can answer Hunt Group's calls from any extensions. An agent must first log in on an extension to answer calls. After the work is done, an agent must log out before leaving. The phone numbers for logging in and out can be set in Special Numbers option. Usually agents will give their code and password for logging in and out. You can set the prompts here:

 Prompts

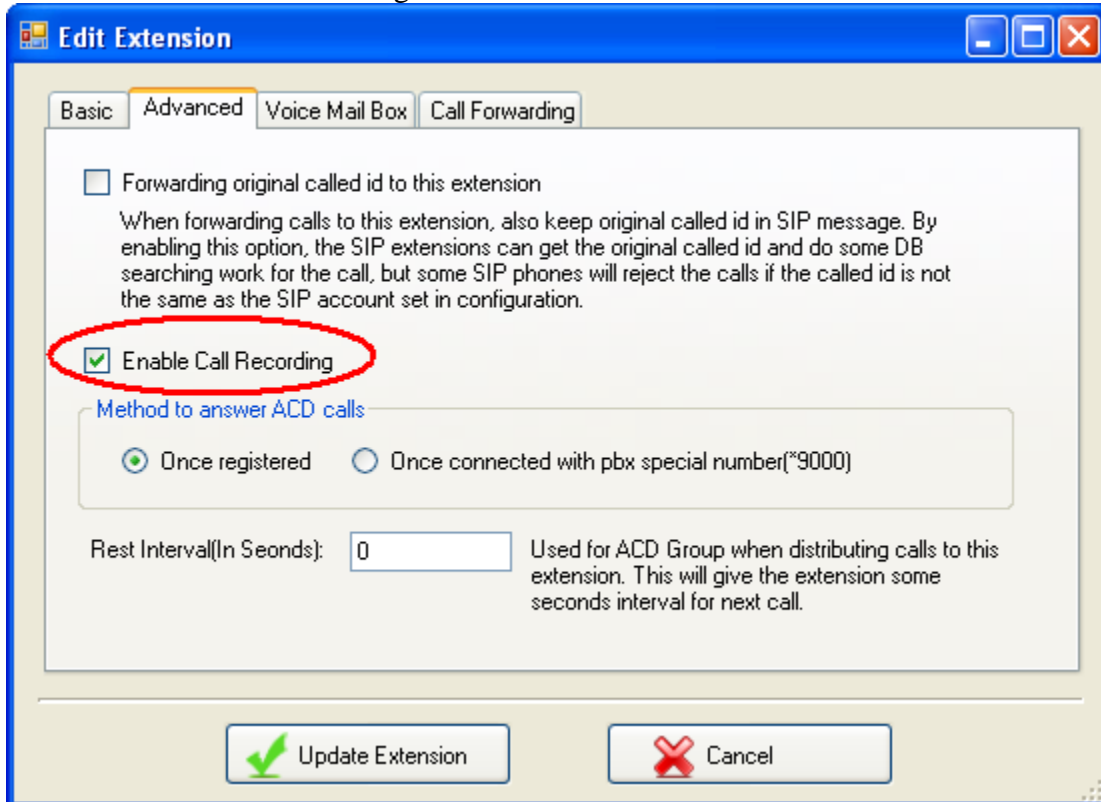
Code	Status	LoginTime
3010	Offline	N/A

 Add  Edit  Delete  OK

4.2 Enabling Call Recording

Also PCBest SIP PBX allows you to record every calls by enabling recording feature for extensions or agents.

Enable extension call recording:



Edit Extension

Basic | **Advanced** | Voice Mail Box | Call Forwarding

☐ Forwarding original called id to this extension
When forwarding calls to this extension, also keep original called id in SIP message. By enabling this option, the SIP extensions can get the original called id and do some DB searching work for the call, but some SIP phones will reject the calls if the called id is not the same as the SIP account set in configuration.

☒ **Enable Call Recording**

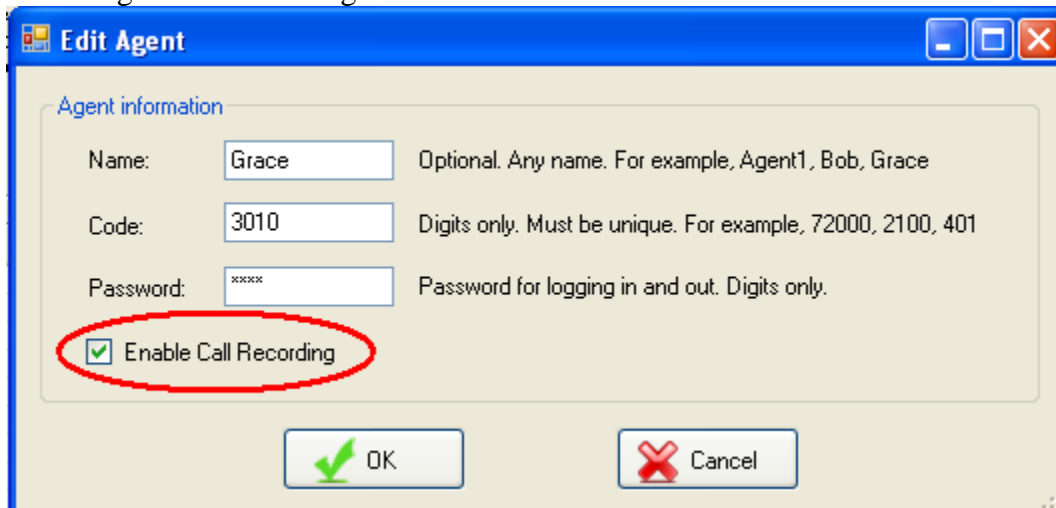
Method to answer ACD calls

☒ Once registered ☐ Once connected with pbx special number(*9000)

Rest Interval(In Seonds): Used for ACD Group when distributing calls to this extension. This will give the extension some seconds interval for next call.

Update Extension Cancel

Enable agent call recording:



Edit Agent

Agent information

Name: Optional. Any name. For example, Agent1, Bob, Grace

Code: Digits only. Must be unique. For example, 72000, 2100, 401

Password: Password for logging in and out. Digits only.

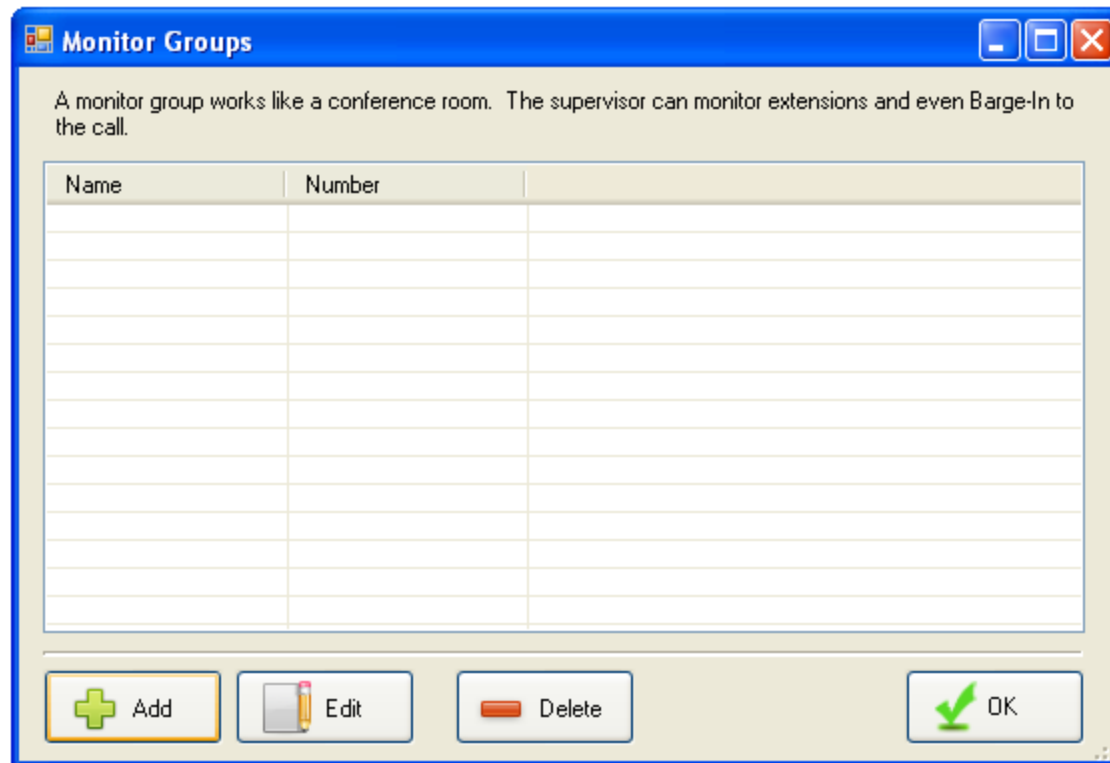
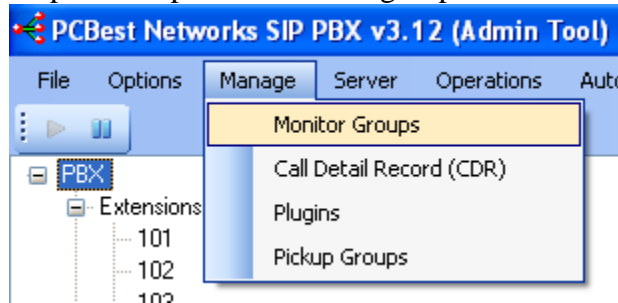
☒ **Enable Call Recording**

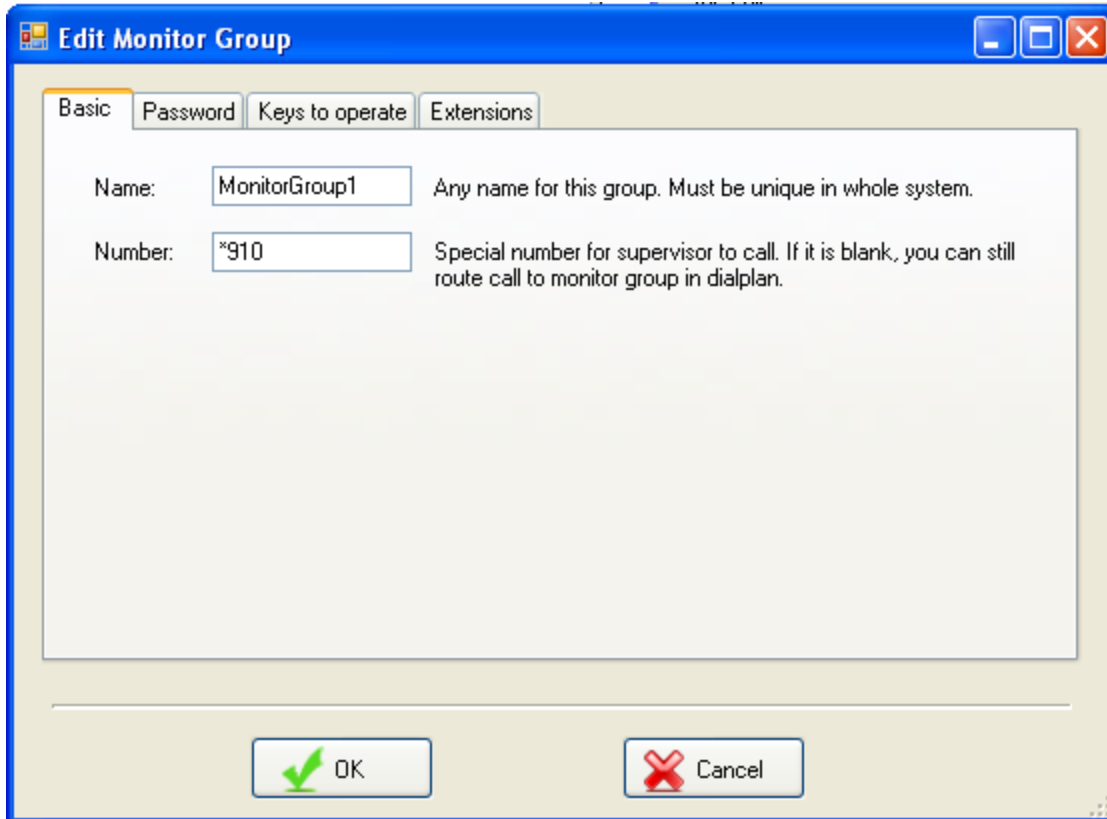
OK Cancel

4.3 Supervisor Call Monitoring

In a typical call centre environment, supervisor needs to monitor agent's call in real time. Sometimes supervisor even can give assistance to agent about how to answer the client's call, or even join into the conversation. In order to achieve the call monitoring, you need to setup a call monitoring group. You can regard a call monitoring group as a conference room, so supervisor, agent and client can all join into.

Steps to setup a call monitor group:





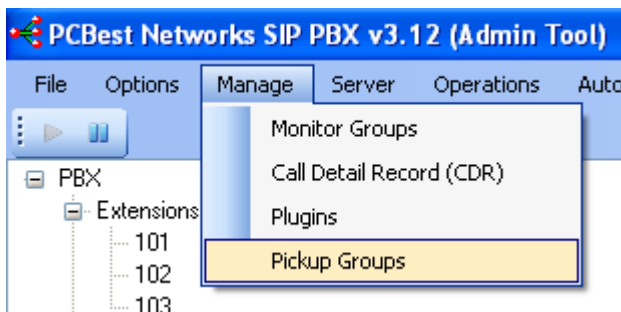
Once you defined a monitor group, please call monitor group number *910 from an supervisor type extension, you will be able to follow the IVR menu to monitor any other extensions.

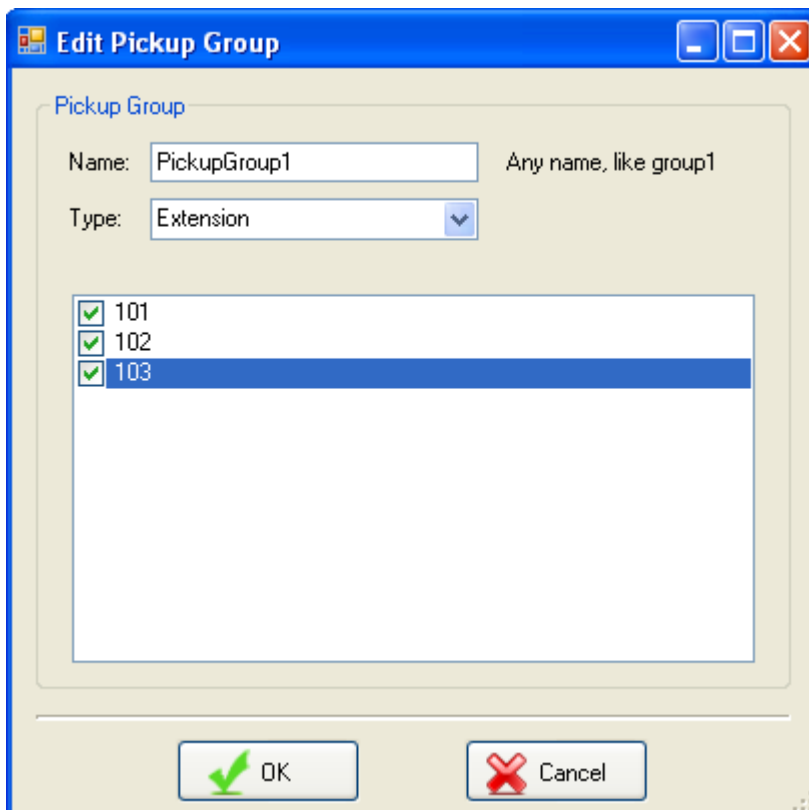
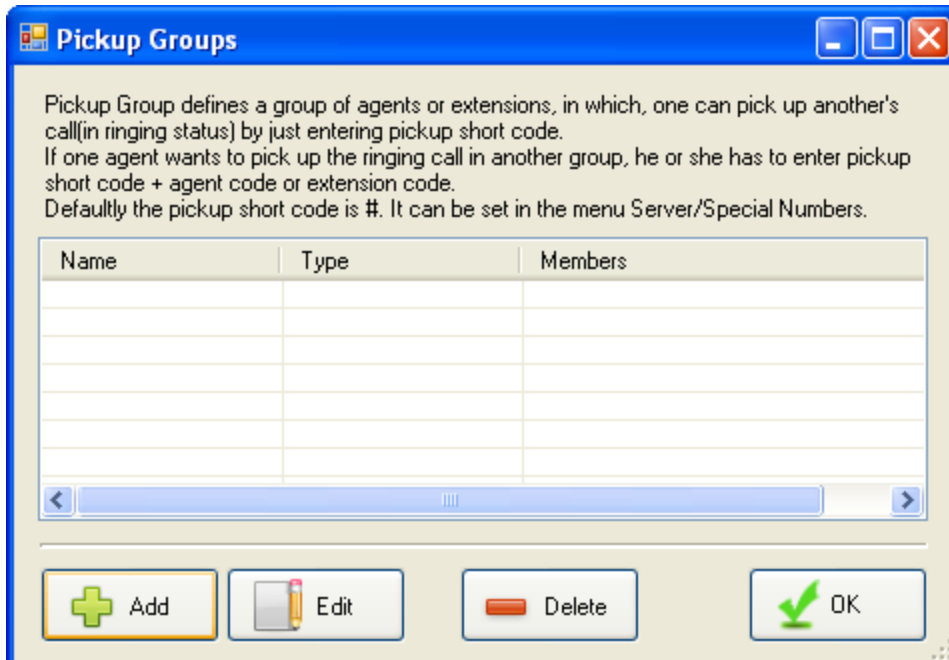
4.4 Pickup Group

Pickup Group defines a group of agents or extensions, in which, one can pick up another's call(in ringing status) by just entering pickup short code.

If one agent wants to pick up the ringing call in another group, he or she has to enter pickup short code + agent code or extension code.

Defaultly the pickup short code is #. It can be set in the menu Server/Special Numbers.





Pickup Group Short Code is defined in special number:

PBX Special Numbers

Basic Magic Transfer Code

Special ACD numbers for extensions to call: *9000

ACD Agent log in number: *71

ACD Agent log out number: *72

Pickup Group Short Code: #

Voice Mail Box Number: *91

OK Cancel

5 PBX Auto Dialer Feature (Pro Only)

PCBest SIP PBX can do automatic outbound calls, and forward connected calls to an inbound dialplan. Auto Dialer Tasks are outbound jobs from database. You can use it to make outbound calls, then do special routes for connected calls. Typical auto dialer tasks can be:

Auto Survey Calls: You can specify an auto dialer task which presents an IVR menu for the connected calls. Once the customer chose an option, then forward the call to another menu, and so on. The customer choices will be record into database like this:

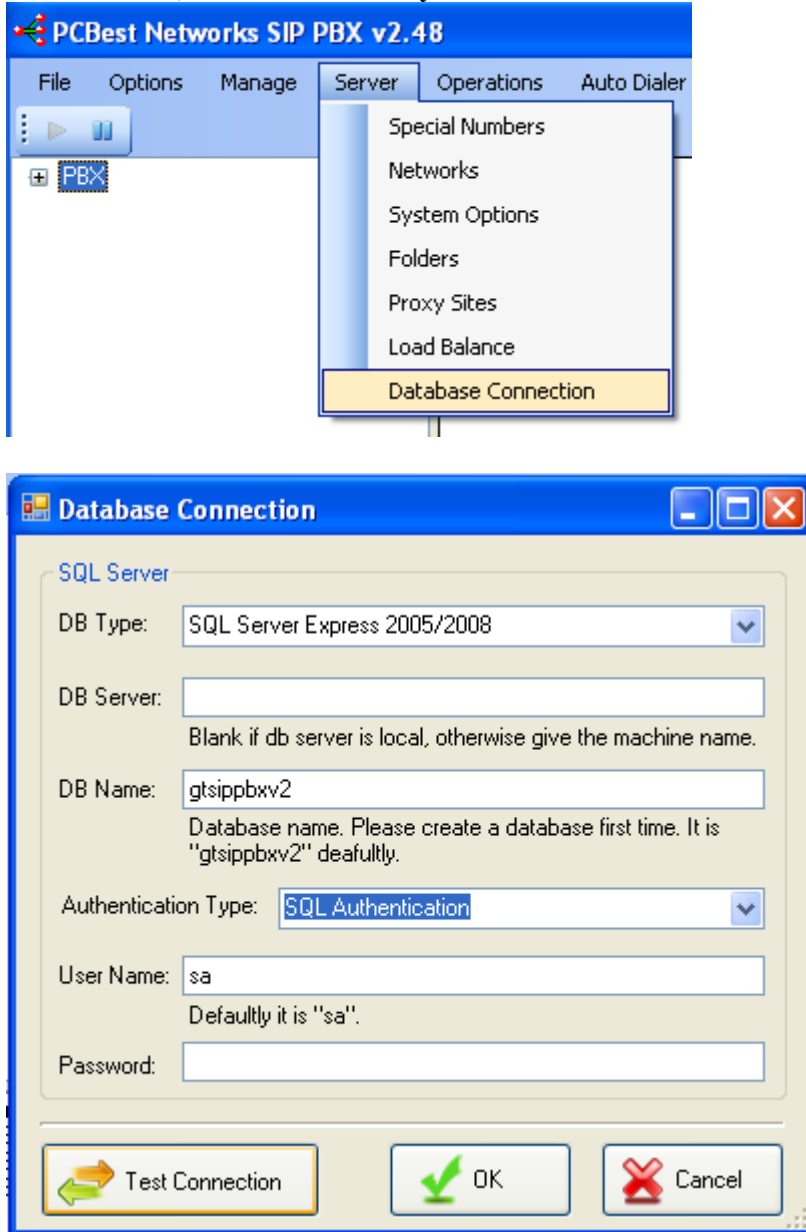
IVRMenu1,1;IVRMenu2,2;...

Call Me Back: Your customer can give a phone number to call back on your website. The phone number will be stored into PBX's auto dialer call jobs table. The pbx will call the number, and once the call is connected, then forward the call to an extension(or agent).

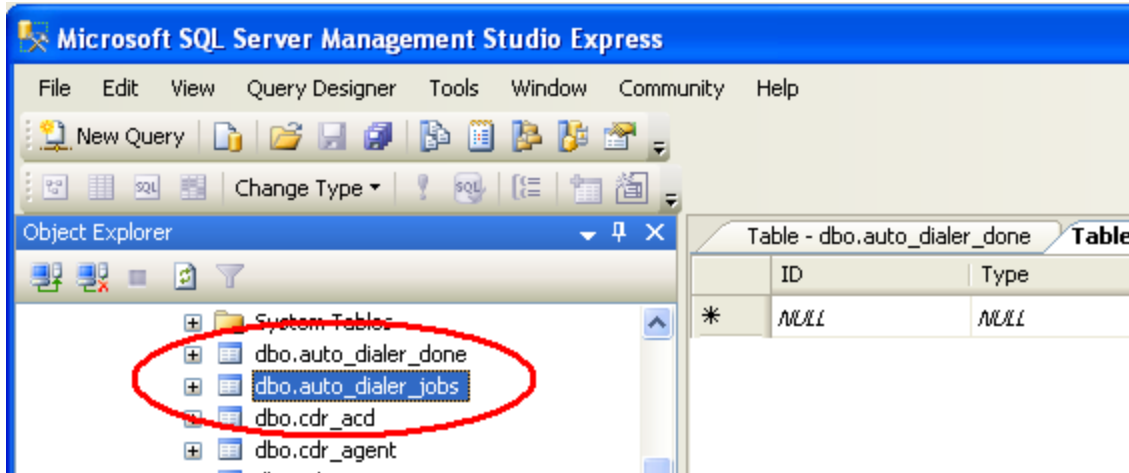
CRM, Message Broadcasting, and other applications: Broadcast your messages to a large of phone numbers to increase your sale.

How does it work?

In order to make this feature works, V2 needs setup a Database Connection. V3 doesn't need, because V3 always works with database.

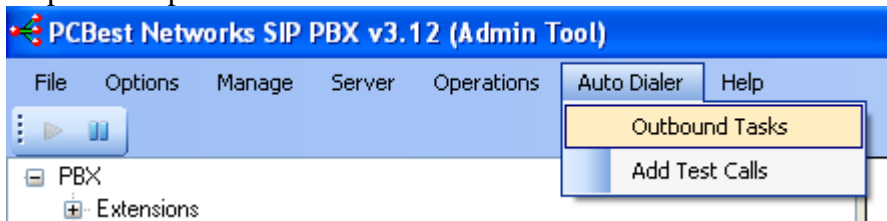


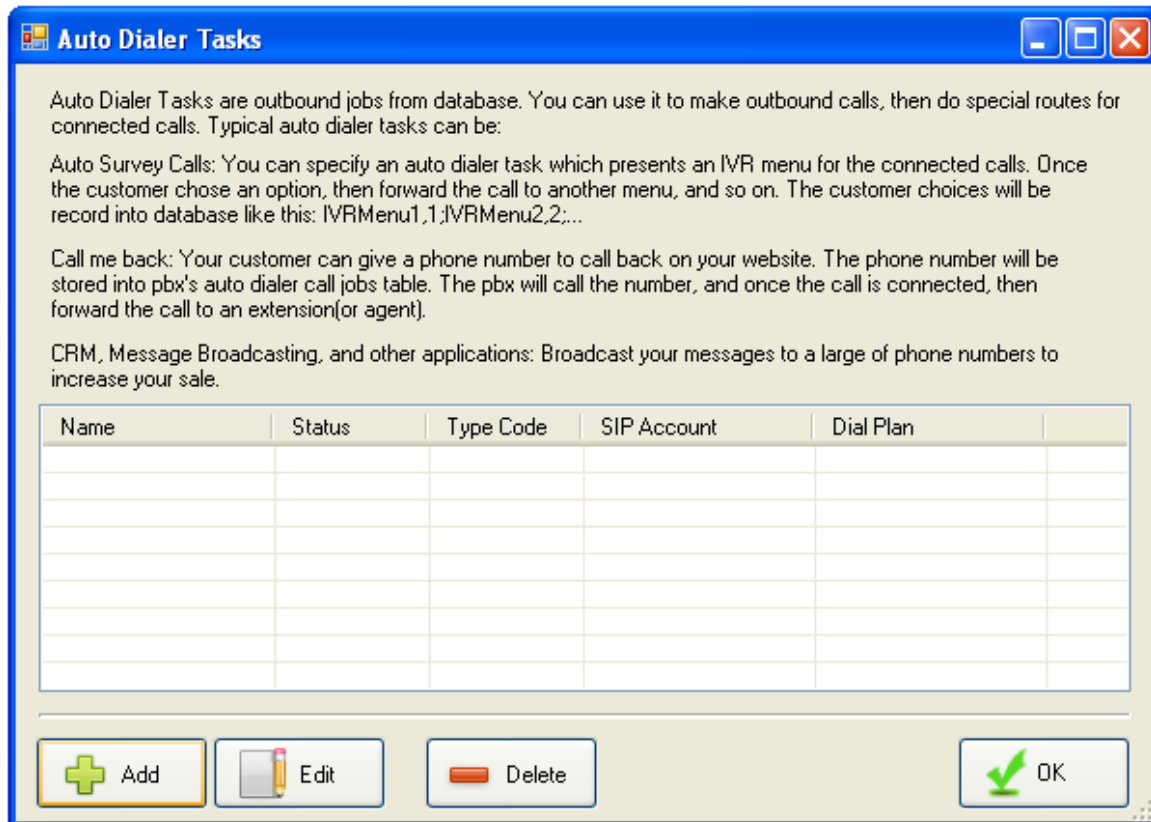
Once the PBX connected with the database, it will create some tables that it needs. Please look at two tables auto_dialer_jobs, and auto_dialer_done.



PBX will try to check auto_dialer_jobs every 2 seconds, to pull out outbound records, then dial the numbers out, then write the result back into auto_dialer_done table.

Steps to setup auto dialer tasks:





Edit Outbound Auto Dialer Task

An outbound task is a group of calls which has the same call action(dialplan).
 You can define as many as outbound tasks you want, but each task must have different type code.
 Each task will pull outbound call jobs, which has the same type code, from auto_dialer_jobs table, and process jobs on idle channels. Once the call is done, it will be saved back into auto_dialer_done table.

Task Info

Name: Any name. For example, Task1, Survey1

☒ Enable this task, so pbx will pick up jobs from database.

Type Code: A small integer code to distinguish tasks in call jobs table(1-32767). This value matches to field 'Type' of auto_dialer_jobs table, and is used to distinguish outbound tasks.

SIP Account: SIP account used to call out

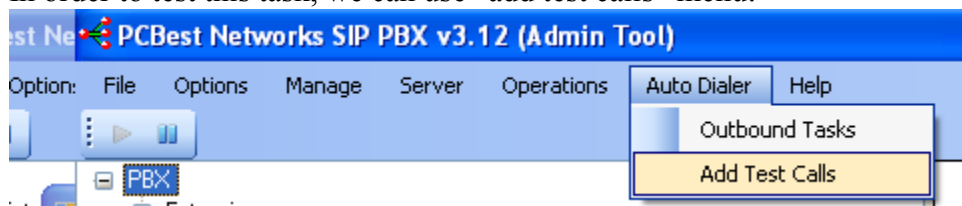
Dial Plan: Inbound dial plan to be used when call is connected.

Stop Ring After: seconds

Max sim calls for this task: 0 means no limit.

Above sample defines auto dialer “Task1”, which has type code 1, and use SIP account “account1” to dial out. After the call is connected, it will use dialplan ToIVR1 to handle the call.

In order to test this task, we can use “add test calls” menu:



Add Test Calls

How many calls to add:

Call Type Code:

Caller:

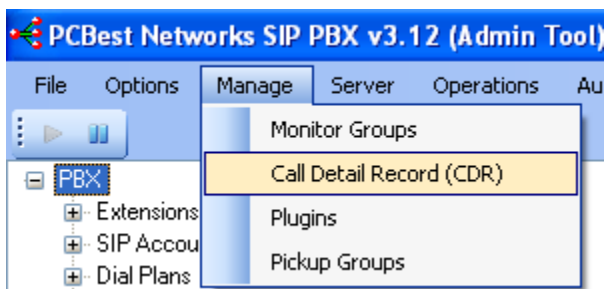
Callee:

Call Start Time:

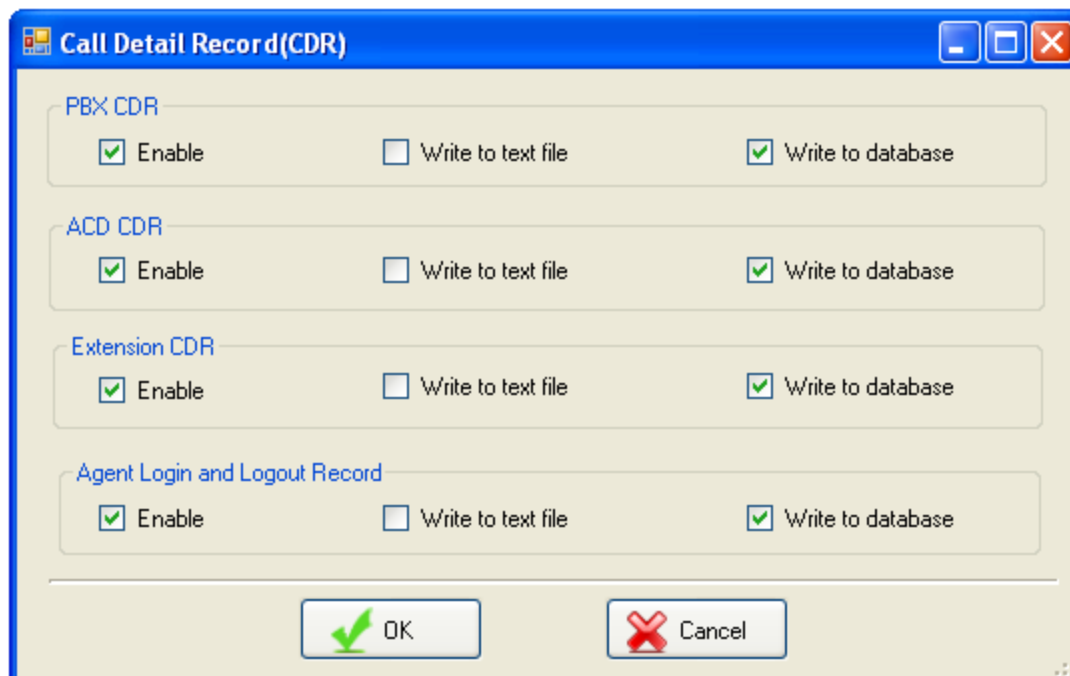
Give the type code 1, and caller and callee numbers, then click OK. PBX should be able to pick up the call job, and dial out to the number.

6 PBX Other Configurations

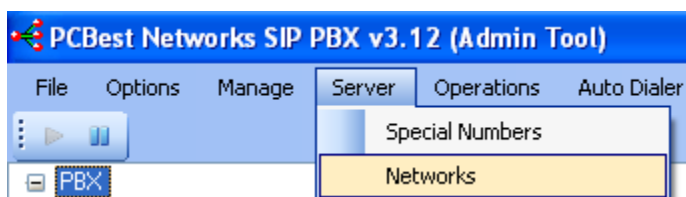
6.1 CDR



You can write CDR into database: (**Note** V2 must setup a database connection first)



6.2 Networks



SIP Networks Tab:

SIP IP Address: The local IP address that PBX should work on. Usually it is blank, so PBX can work on all possible NIC interfaces or IPs. If you do have multiple IP addresses, and want the PBX only work on one of them, please use drop box to select.

SIP Port: The port number that PBX works on for SIP protocol. Default it is 5060, but you can change it something else. For example, some countries block 5060 to disable VoIP calls. You can use other port number to get around.

RTP Port From: The starting RTP port number. Sometimes you may need to open your firewall for RTP(audio) transmit. Keep in mind, PBX will use a range of RTP port for communication. Basically one channel will use 4 ports(although it only use actually one, but we separate them with enough space), so one 8 channels PBX will need ports open from 19200 to 19232 ($19200 + 4 \times 8$).

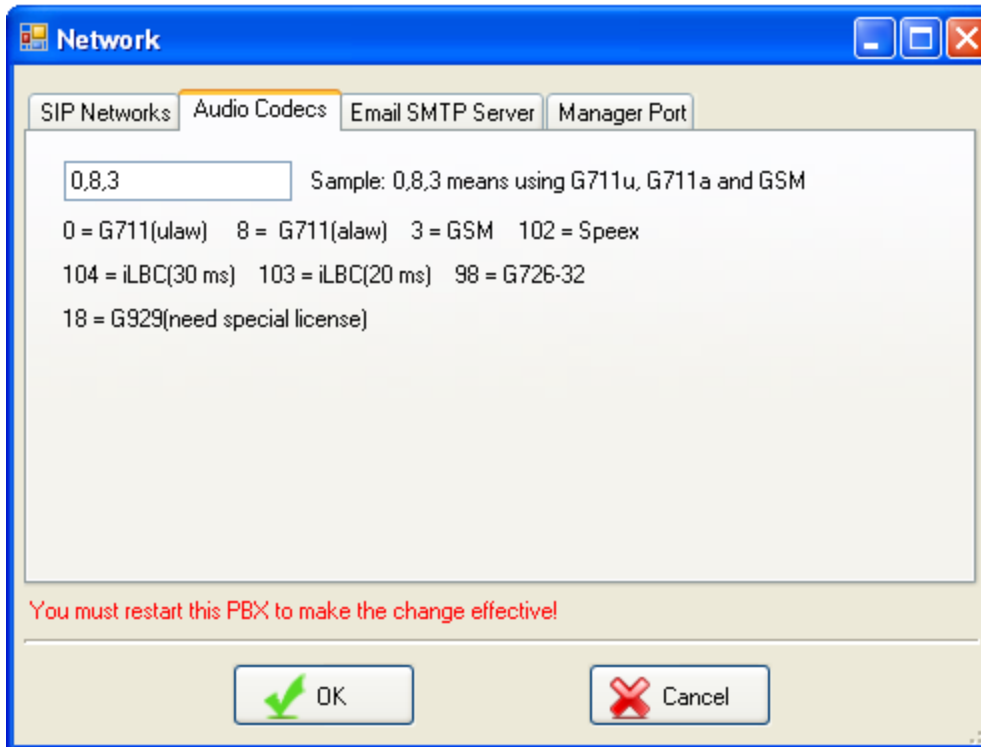
Internal: PBX uses this internal port for internal messages and events communication. It is not changeable.

STUN Server: PBX uses STUN server to discover the actual public IP address of network, to go through possible NAT issue. Please contact your SIP service provider for STUN server setting.

DTMF Method: Usually it is auto, so PBX will automatically figure out the DTMF method. Unless you know the details about this setting, you can change it.

Public IP Address(V3 Only): In some case, for example, DMZ, you know your PBX are working on specific public IP address, so you can specify this field so PBX won't use internal IP address or ignore STUN server to get public IP.

Audio Codec Tab:



You can specify the PBX which audio codec in SIP SDP negotiation. When negotiating the audio codec, PBX will try to use the audio codec that is in the front of the list. In above sample, the audio codec is 0,8,3. It means that g711 mulaw first, then g711 alaw, then GSM.

Email SMTP Server:

Network

SIP Networks Audio Codecs **Email SMTP Server** Manager Port

PBX will use this email account to send voice mail to individual's email address.

Server: Sample: mail.abc.com,123.67.9.67

Port: Default: 25

Email: Sample: abc@gmail.com

Password:

☐ Enable SSL

You must restart this PBX to make the change effective!

Server: Email server address. It can be an IP address or domain name.

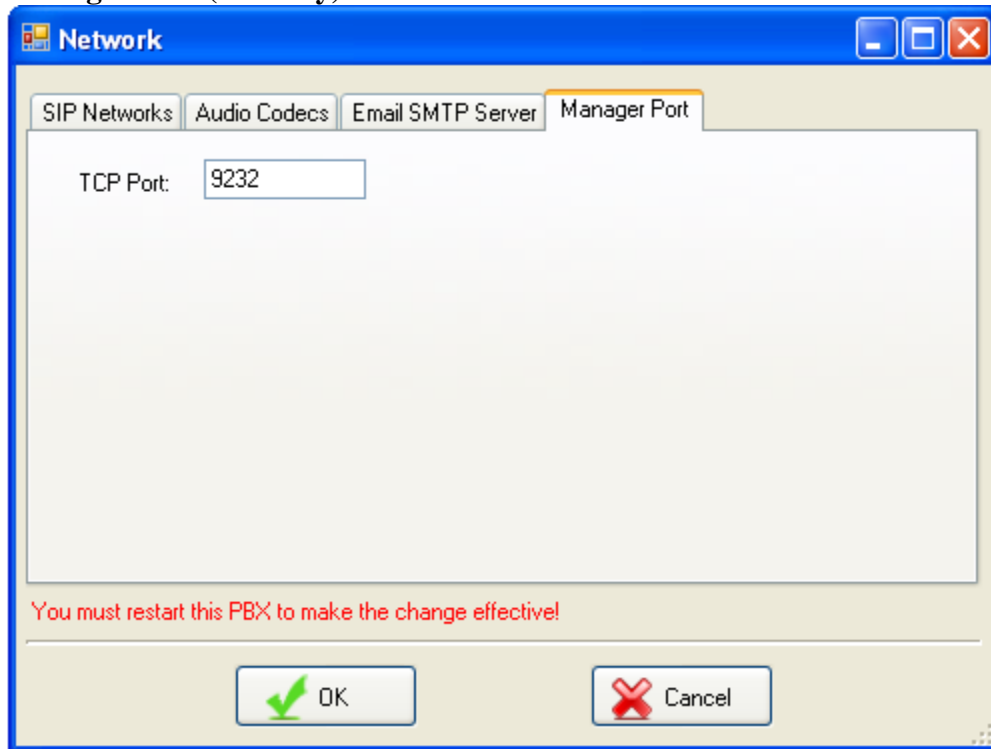
Port: Email server port number.

Email: Email address that is used by PBX to send out email.

Password: Password for above email address.

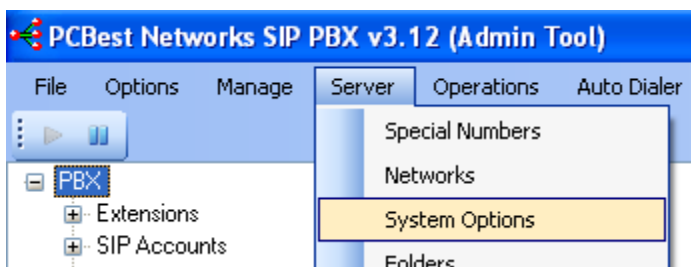
Enable SSL: if it uses SSL.

Manager Port (V3 Only):

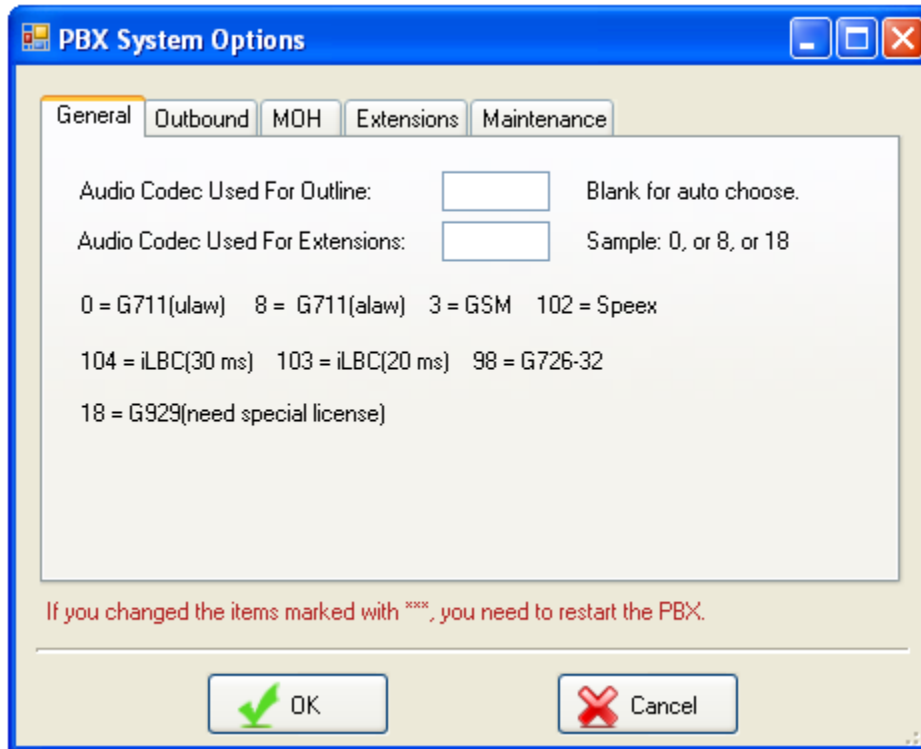


Manager port is used to for manager client to connect. PBX has a sample in SDK named “ManagerClient”, which shows how to develop .NET application to receive events from PBX, or control PBX. Please refer to 7.2 about details.

6.3 System Options



General Tab:



A typical example is that sometimes, you want low bandwidth audio codec using on the public network, but high quality audio codec on intranet.

Above dialog give you an option to specify the outline codec and internal codec.

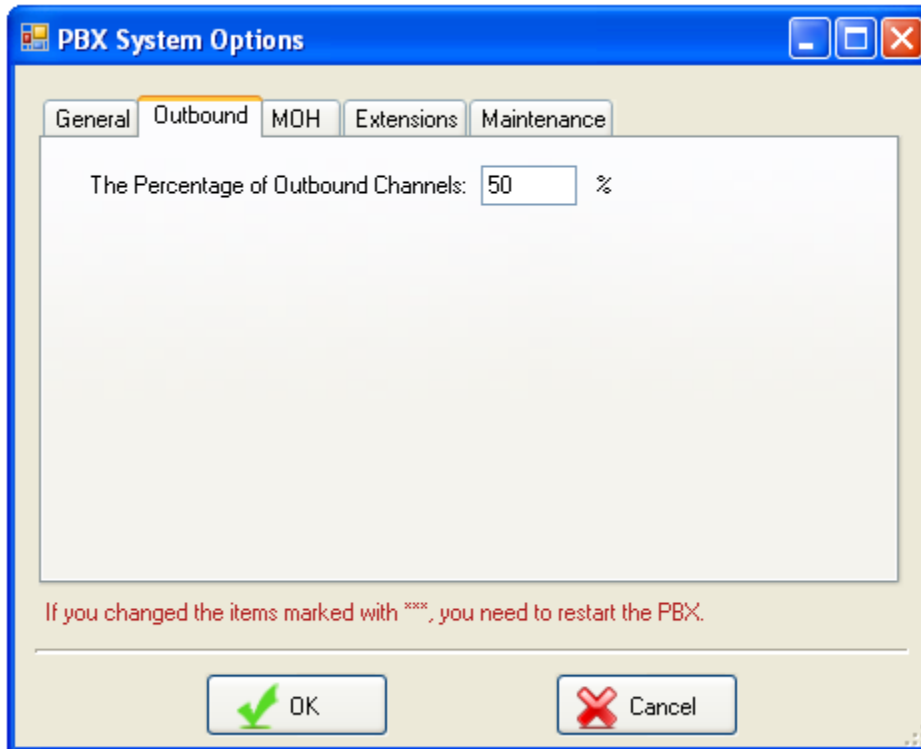
For example, you can specify:

Outline: 18

Extensions: 0

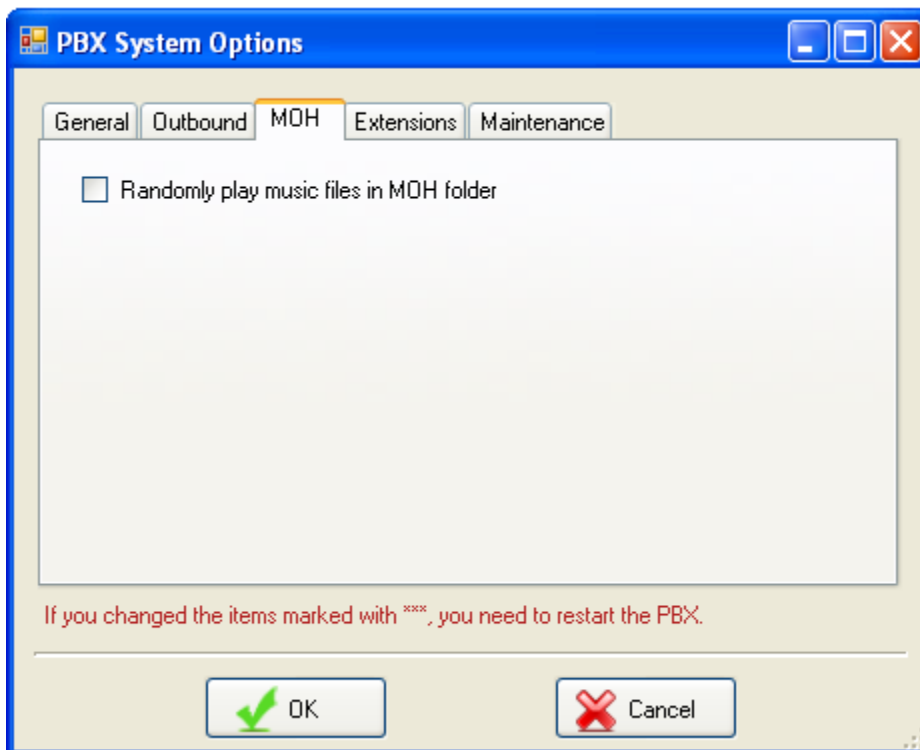
It means PBX will do audio codec converting from g711 to g729 when extension calls out. In another word, PBX will use g711 to handle extension calls, and use g729 for outline.

Outbound Tab:

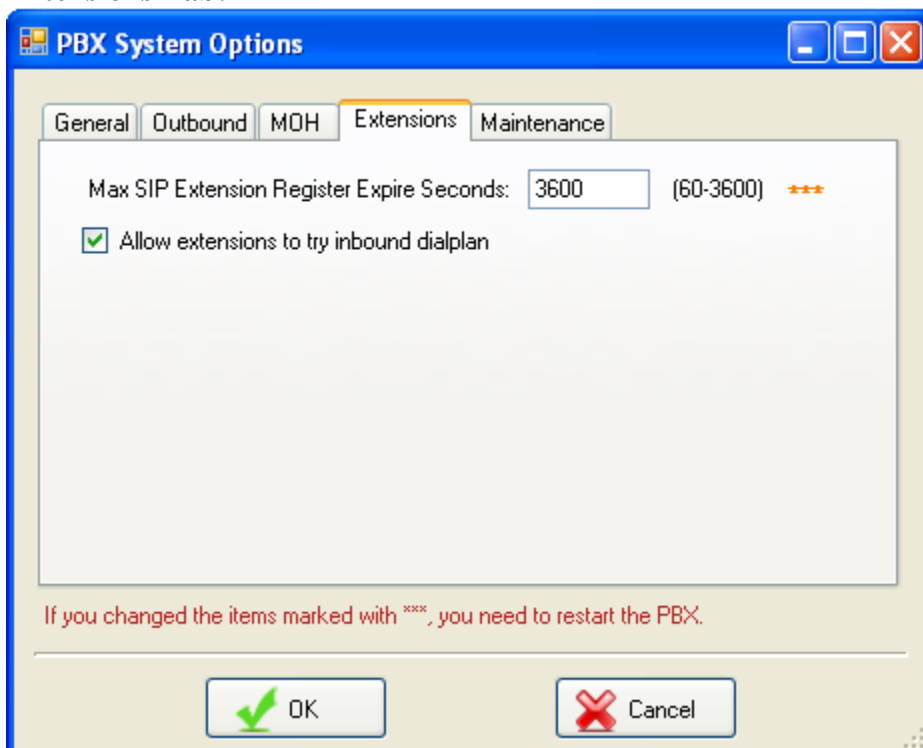


Percentage of outbound channels is for outbound calls. PBX default uses half channels for outbound, and keep half channels for inbound calls.

MOH Tab:



Extensions Tab:

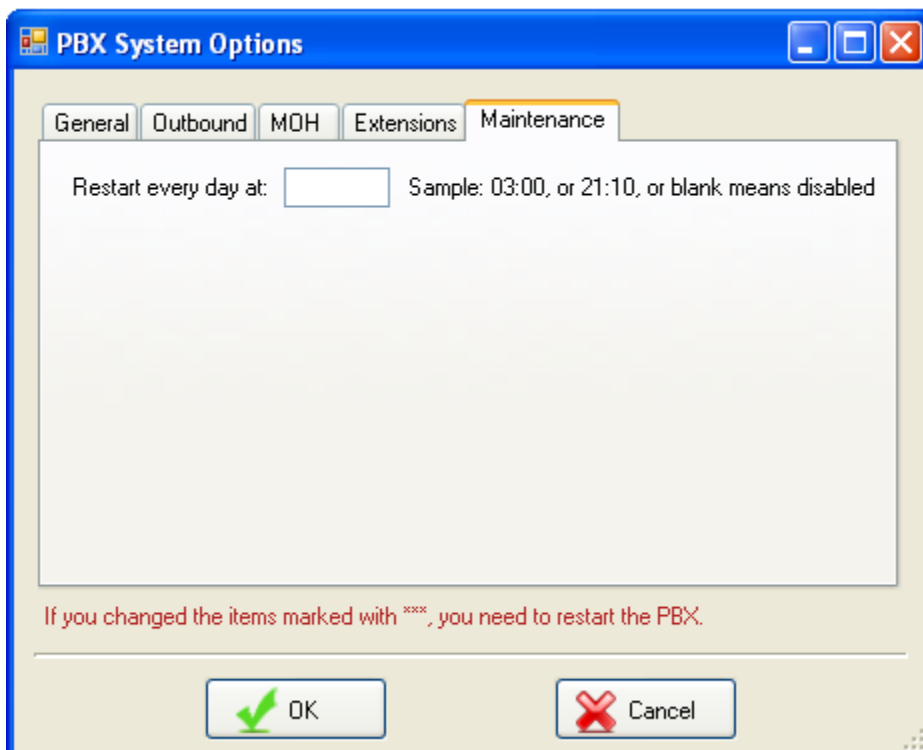


The screenshot shows the 'PBX System Options' window with the 'Extensions' tab selected. The 'General' tab is also visible. The 'Extensions' tab contains the following settings:

- Max SIP Extension Register Expire Seconds: 3600 (60-3600) ***
- ☒ Allow extensions to try inbound dialplan

At the bottom of the window, there is a red text warning: "If you changed the items marked with ***, you need to restart the PBX." Below this warning are two buttons: 'OK' (with a green checkmark icon) and 'Cancel' (with a red X icon).

Maintenance:

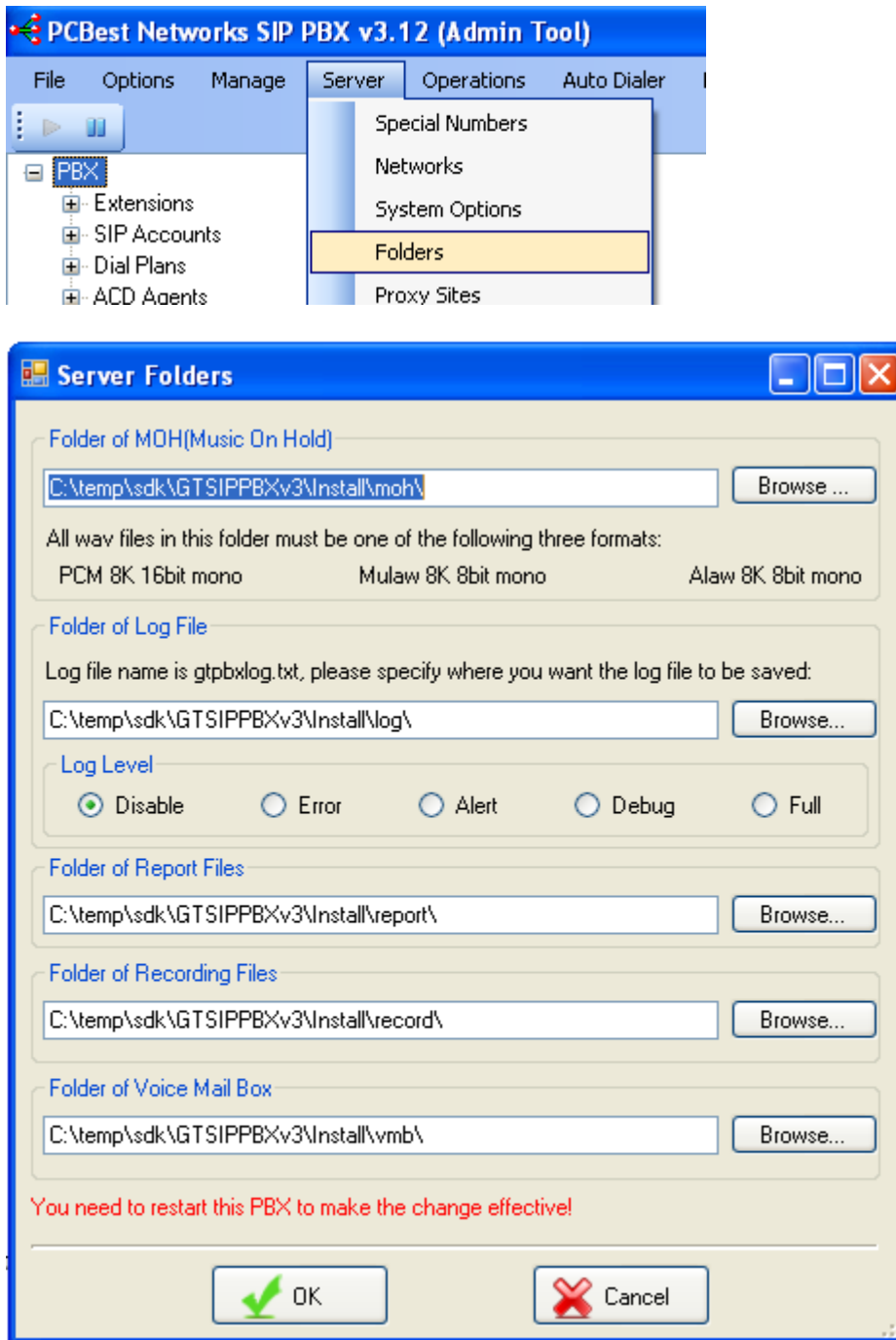


The screenshot shows the 'PBX System Options' window with the 'Maintenance' tab selected. The 'General' tab is also visible. The 'Maintenance' tab contains the following settings:

- Restart every day at: [] Sample: 03:00, or 21:10, or blank means disabled

At the bottom of the window, there is a red text warning: "If you changed the items marked with ***, you need to restart the PBX." Below this warning are two buttons: 'OK' (with a green checkmark icon) and 'Cancel' (with a red X icon).

6.4 Folders and Logs



Enable Log: Please set log level to Full, and after restarting PBX, you should be able to find log files in log folder.

7 PBX Developments (Pro Only)

7.1 Plugin

Plugin allows you extend PBX's feature. Plugins are external dlls that exist in "plugin" sub folder, and are loaded when PBX starts. There are three C# plugin samples in pbx plugin sub folder, for three types of PBX plugins, "IVRMenu", "Init" and "Routine".

"IVRMenu" plugin is used to extend PBX's IVR feature. It allows you customize your own IVR menu, or do your special routes before it reaches extensions.

"Init" plugin is executed when pbx starts and stops. For example, you can use your own data from DB to set PBX parameters.

"Routine" plugin runs every one second, to let you do your own job for special purpose. For example, restarting PBX regularly, or adding more extensions.

There are also five call states plugin, when the call state changed.

"CallIdle"

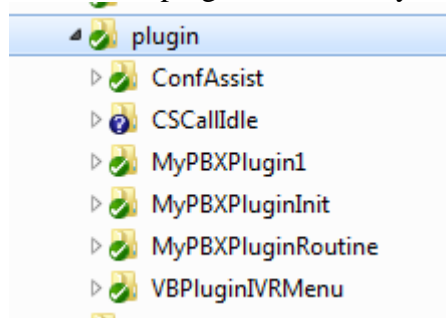
"CallOffered"

"CallRinging"

"CallDialing"

"CallConnected"

In the PBX plugin sub folder, you can find samples of plugin.



MyPBXPlugin1 is a sample for IVRMenu type plugin. C# code.

MyPBXPluginInit is a sample for Init type plugin. C# code.

MyPBXPluginRoutine is a sample for Routine type plugin. C# code.

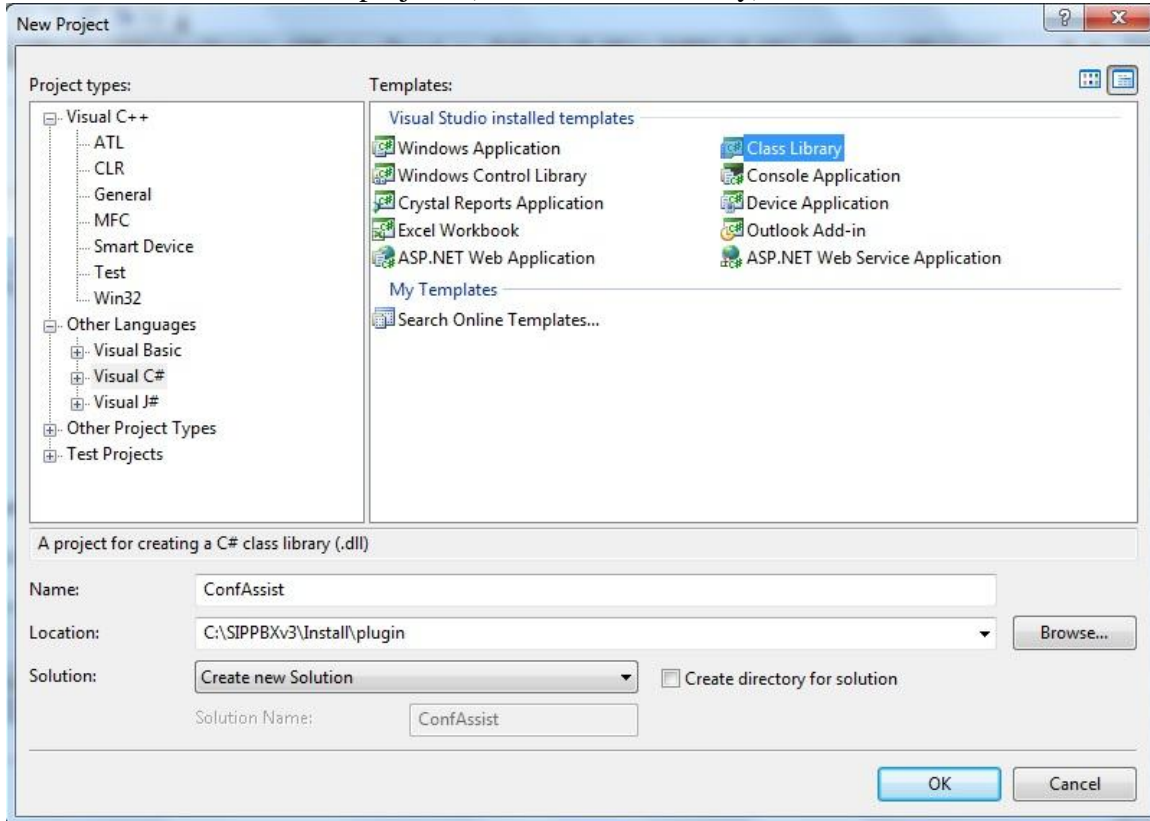
VBPluginIVRMenu is a sample for IVRMenu type plugin in vb.net code.

CSCallIdle is a sample for call idle type plugin. C# code.

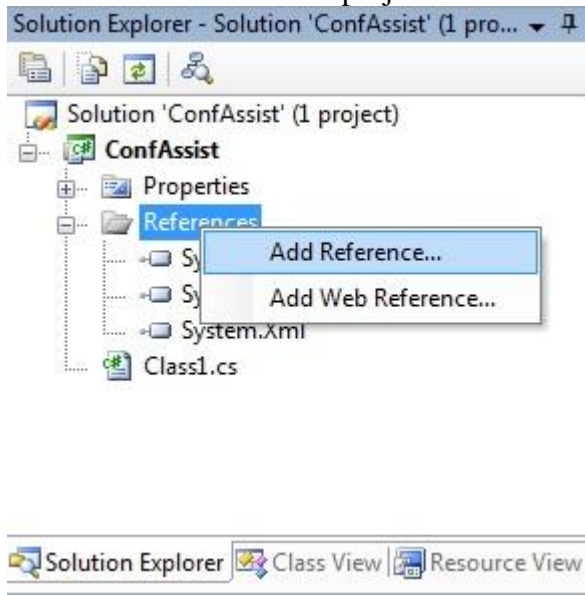
ConfAssist is a sample for IVRMenu type plugin which call advanced conference functions. C# code.

How to setup a plugin project?

1. New a vb.net or C# project: (Choose Class Library)

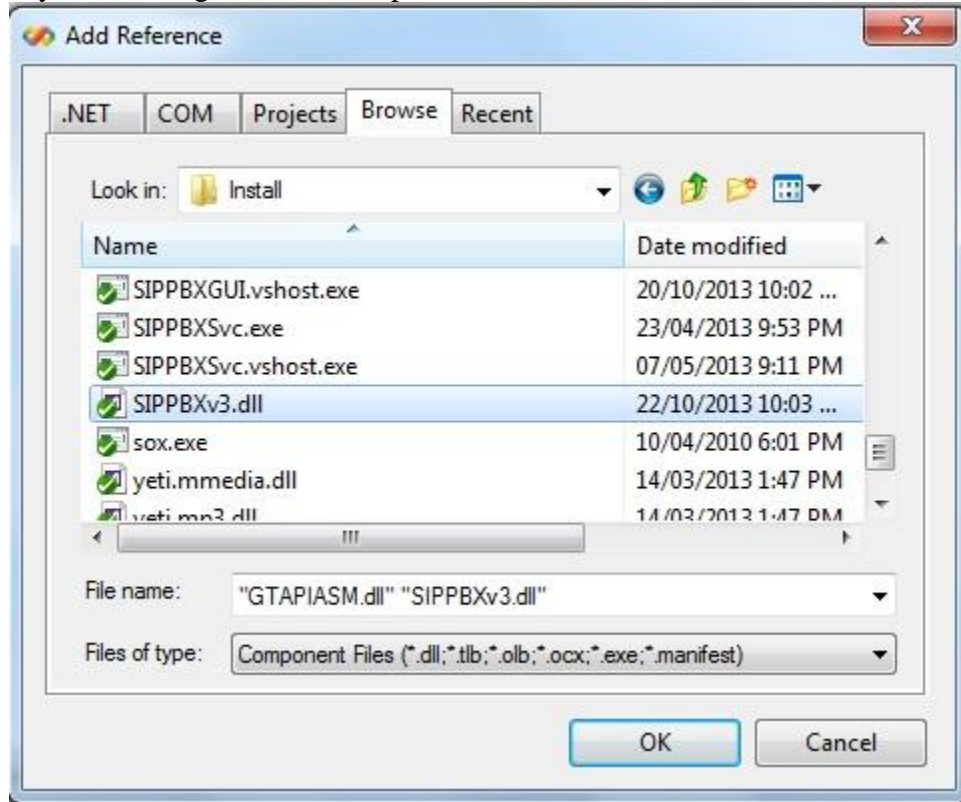


2. Then add reference to the project:



If you are using SIP PBX v2, please choose GTSIPPBX.exe in pbx installation folder.

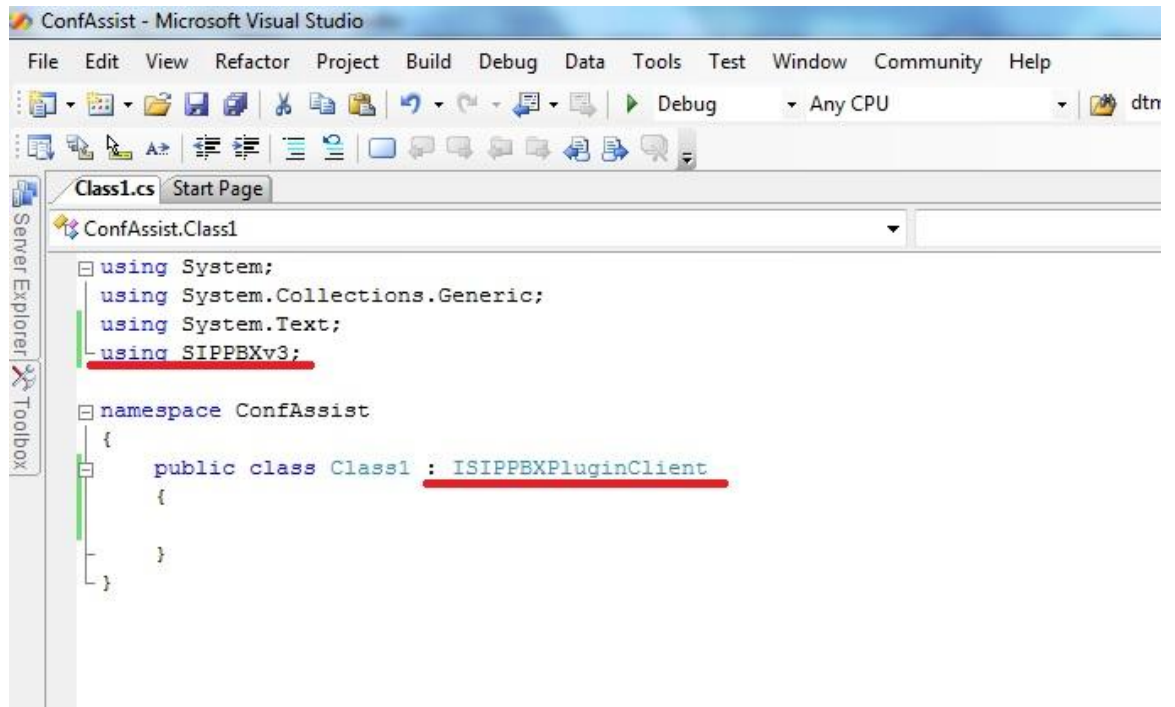
If you are using SIP PBX v3, please choose SIPPBXv3.dll.



Also, you need add GTAPIASM.dll as reference.

Note: DO NOT forget to add reference "System.Windows.Forms".

3. Write a class which implements interface `ISIPPBXPluginClient`



```
public void Start()
{
    //get the caller and callee number
    string caller_addr = Host.Channel.caller_num;
    string callee_addr = Host.Channel.callee_num;

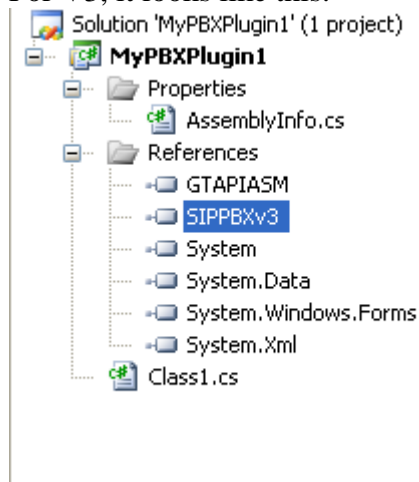
    string caller_num = GTAPIASM.GTAPIEnv.GetSIPAddressInfo(1, caller_addr);
    string callee_num = GTAPIASM.GTAPIEnv.GetSIPAddressInfo(1, callee_addr);

    //if it is a job of auto dialer task for human/answer machine detection,
    //use the following code to access detection result:
    /*
    if (Host.PBX_Channel.call_job != null)
    {
        switch (Host.PBX_Channel.call_job.DetectResult)
        {
            case 0: // = Answering Machine
                break;
        }
    }
    */
}
```

Please refer to C# or VB.NET plugin sample code for this part.

Let us open MyPBXPlugin1.

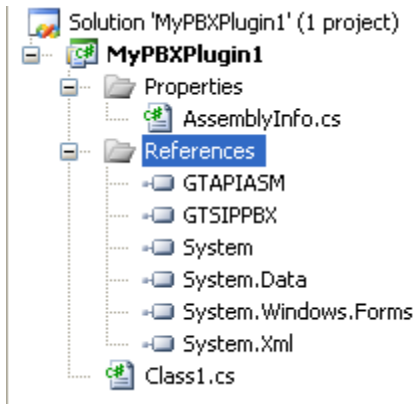
For V3, it looks like this:



Please change the references about GTAPIASM, and SIPPBXv3 if they are not available and pointing to right dlls.

SIPPBXv3 is SIPPBXv3.dll.

If you are using V2, it should looks like this:



GTSIPPBX refers to V2's GTSIPPBX.exe.

Please open class1.cs for less than 200 lines sample, which teaches you how to write the plugin IVR sample.

Plugin Built-in Methods and Functions

Host.DisplayMenu

Display a menu, and accept DTMF inputs.

Format: `string DisplayMenu(string audio_fn, int maxDigits, string termStr, int timeout);`

audio_fn: Audio file name in full path.

maxDigits: The maximum digits to accept for the menu.

termStr: The string contains the digit which terminate the DTMF inputs. In most of cases, it is "#".

timeout: how long to wait. In milliseconds.

Return: DTMF string

Host.DisplayMenuEx

Display a multiple-audio menu, and accept DTMF inputs.

Format: `string DisplayMenu(List<string> audio_files, int maxDigits, string termStr, int timeout);`

audio_files: Audio files in full path to be played.

maxDigits: The maximum digits to accept for the menu.

termStr: The string contains the digit which terminate the DTMF inputs. In most of cases, it is "#".

timeout: how long to wait. In milliseconds.

Return: DTMF string

Host.PlayAudio

Play an audio file.

Format: `string PlayAudio(string audio_fn, int maxDigits, string termStr, int timeout);`

audio_fn: Audio file name in full path.
maxDigits: The maximum digits to accept for the menu.
termStr: The string contains the digit which terminate the DTMF inputs.
In most of cases, it is "#".
timeOut: how long to wait. In milliseconds.

Return: DTMF string

Host.PlayAudioEx

Play an audio file.

Format: `string PlayAudioEx(List<string> audio_files, int maxDigits, string termStr, int timeOut);`
audio_files: Audio files in full path to be played.
maxDigits: The maximum digits to accept for the menu.
termStr: The string contains the digit which terminate the DTMF inputs.
In most of cases, it is "#".
timeOut: how long to wait. In milliseconds.

Return: DTMF string

Host.RecordAudio

Record an audio file.

Format: `string RecordAudio(string audio_fn, int maxDigits, string termStr, int timeOut);`
audio_fn: Audio file name in full path.
maxDigits: The maximum digits to accept for the menu.
termStr: The string contains the digit which terminate the DTMF inputs.
In most of cases, it is "#".
timeOut: how long to wait. In milliseconds.

Return: DTMF string

Host.DetectDTMF

Detect DTMF keys.

Format: `string DetectDTMF(int maxDigits, string termStr, int timeOut);`
maxDigits: The maximum digits to accept for the menu.
termStr: The string contains the digit which terminate the DTMF inputs.
In most of cases, it is "#".
timeOut: how long to wait. In milliseconds.

Return: DTMF string

Host.HangUp

Disconnect call.

Format: `int HangUp();`

Return: none

Host.WriteLog

Write a log information in the PBX GUI output and log.

Format: `WriteLog(string logInfo);`
logInfo: the log text.

Return: none

Host.ToExtension

Transfer this call to extension.

Format: `bool ToExtension(string exten_no);`
exten_no: the extension number

Return: bool, if succeed.

Host.ToIVRMenu

Send this call to IVR menu.

Format: `bool ToIVRMenu(string menu_name);`
menu_name: the IVR menu name defined in PBX.

Return: bool, if succeed.

Host.ToMonitorGroup

Send this call to monitor group.

Format: `bool ToMonitorGroup(string mg_name);`
mg_name: the monitor group name defined in PBX.

Return: bool, if succeed.

Host.ToHuntGroup

Send this call to ACD group.

Format: `bool ToHuntGroup(string acd_name, bool set_front);`
acd_name: the hunt group name defined in PBX.
set_front: if set the call to the front of group so it can be answered immediately.

Return: bool, if succeed.

Host.ToRingGroup

Send this call to ring group.

Format: `bool ToRingGroup(string rg_name);`
rg_name: the ring group name defined in PBX.

Return: bool, if succeed.

Host.ToVoiceMailBox

Send this call to voice mail box.

Format: `bool ToVoiceMailBox(string exten_no);`
exten_no: the extension number of which voice mail box being used.

Return: bool, if succeed.

Host.ToConferenceRoom

Send this call to a conference room.

Format: `bool ToConferenceRoom(string conf_name);`
conf_name: conference room name defined in PBX.

Return: bool, if succeed.

Host.ToPlugin

Send this call to another plugin.

Format: `bool ToPlugin(string plugin_name);`
plugin_name: another plugin's name.

Return: bool, if succeed.

Host.ToNumber

Forward this call to another phone number.

Format: `bool ToNumber(string number, SIPAccount sip_acct);`
number: the number to forward.
sip_acct: sip account to use for this call

Return: bool, if succeed.

Sample:

```
//to another outside number
//Host.ToNumber("<sip:123@192.168.1.100>", null);
//Host.ToNumber("<sip:6781992@callcentric.com>", null);
//or
//SIPAccount acct1;
//acct1.DisplayName = "any";
//acct1.UserName = "1234";
//acct1.DomainServer = "sip.callwithus.com";
//acct1.ProxyServer = "sip.callwithus.com";
//acct1.AuthName = "1234";
```

```
//acct1.Password = "xxxxx";  
//Host.ToNumber("655112", acct1);
```

Host.DisconnectExtension

Disconnect(hang up) extension's call.

Format: `bool DisconnectExtension(string exten_no);`
exten_no: extension number.

Return: bool, if succeed.

Host.SetChanRunPlugin

Set channel to run another plugin.

Format: `bool SetChanRunPlugin(int ch, string plugin_name);`
ch: channel number.
plugin_name: the name of plugin.

Return: bool, if succeed.

Host.ResetChannel

Reset the channel. Disconnect the call if there is a call on the channel.

Format: `bool ResetChannel(int ch);`
ch: channel number.

Return: bool, if succeed.

Host.SetChanInConferenceRoom

Set channel into conference room.

Format: `bool SetChanInConferenceRoom(int ch, string conf_name, int opt);`
ch: channel number.
conf_name: conference room.
opt: 0 = take out of conference room. 1 = add into conference room. 2 = monitor(listening only, not speaking)

Return: bool, if succeed.

Host.CreateConferenceRoom

Create a conference room

Format: `SIPConferRoom CreateConferenceRoom(string conf_name);`
conf_name: conference room.

Return: bool, if succeed.

Host.DestroyConferenceRoom

Destroy a conference room

Format: `void DestroyConferenceRoom(string conf_name);`
conf_name: conference room.

Return: bool, if succeed.

Host.GetConferenceRoomIndex

Get conference room index.

Format: `int GetConferenceRoomIndex(string conf_name);`
conf_name: conference room.

Return: the index of conference room.

Host.GetConferenceRoomHandle

Get conference room handle.

Format: `ulong GetConferenceRoomHandle(string conf_name);`
conf_name: conference room.

Return: the handle of conference room.

Host.GetConferenceRoomByName

Get conference room handle.

Format: `SIPConferRoom GetConferenceRoomByName(string conf_name);`
conf_name: conference room.

Return: the class of conference room.

Host.SetUserObj

Set user object for application, in order to retrieve it later

Format: `bool SetUserObj(int idx, object obj);`
idx: index of the object, based on 0.
obj: the object.

Return: if succeed.

Host.GetUserObj

Set user object for application, in order to retrieve it later

Format: `object GetUserObj(int idx);`
idx: index of the object, based on 0.

Return: the object

Host.GetChanUserObj

Get channel's object

Format: `object GetChanUserObj(int ch, int idx);`
ch: channel index based on 0.
idx: index of the object

Return: the object

Host.SetChanUserObj

Set channel's object

Format: `bool SetChanUserObj(int ch, int idx, object obj);`
ch: channel index based on 0.
idx: index of the object
obj: object

Return: if succeed.

Host.StartPBX

Start PBX

Format: `void StartPBX();`

Return: none.

Host.StopPBX

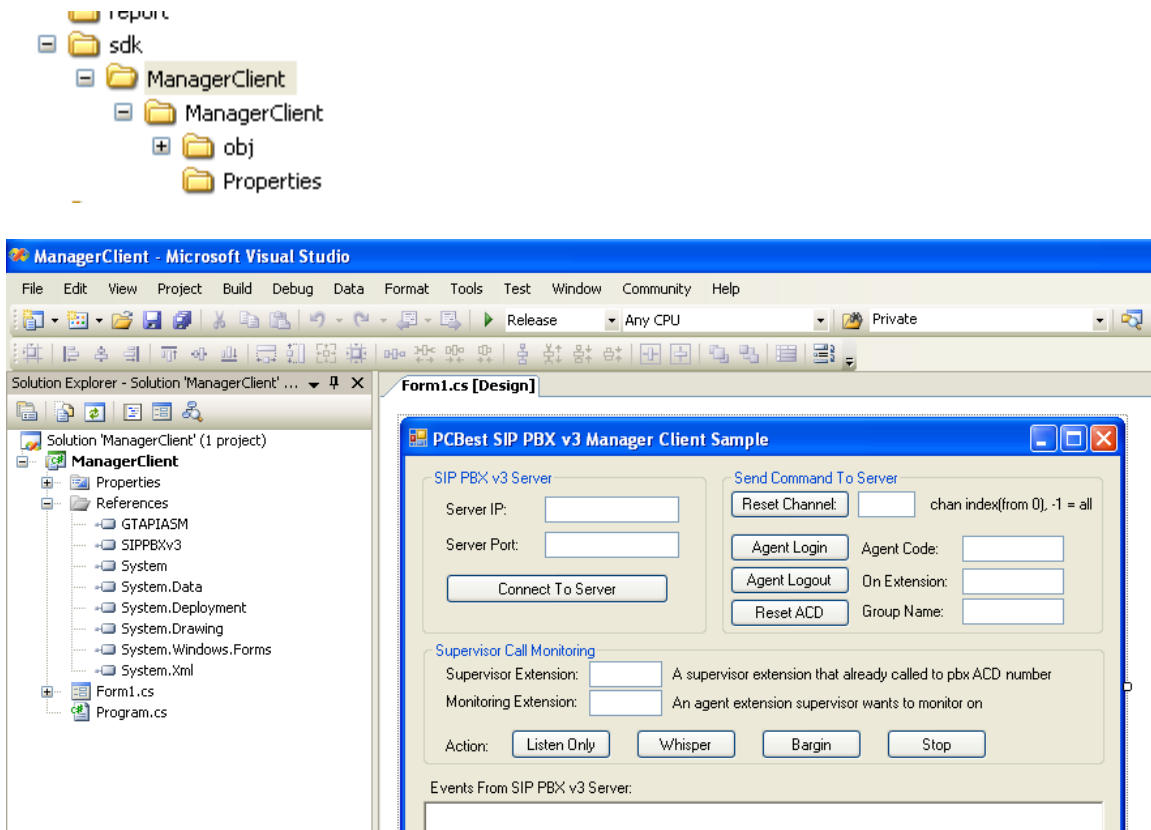
Stop PBX

Format: `void StopPBX();`

Return: none

7.2 Manager Client Application (V3 only)

Manager client application is used for agent desktop computer to receive additional call information, or manager to control the PBX. Please open PBX SDK subfolder, you will see the a full source code of manage client application.



Currently manage client can receive those events:

Call events on each channel.

Registration events of extensions.

Agent login and logout event.

Manage client can also do those actions:

1. ***Connect/Disconnect to PBX server.***
2. ***Reset channels.***
3. ***Reset ACD group.***
4. ***Agent login and logout.***
5. ***Supervisor monitors extension.***
6. ***Dial a number for extension***
7. ***Make, Answer, and Hang up call on specific channel***
8. ***Hold and Transfer call on specific channel***
9. ***Run plug-in on the specific channel***
10. ***Do magic transfer for specific channel***
11. ***Extension status, Channel Status, and Agent Status events.***

Methods and Events

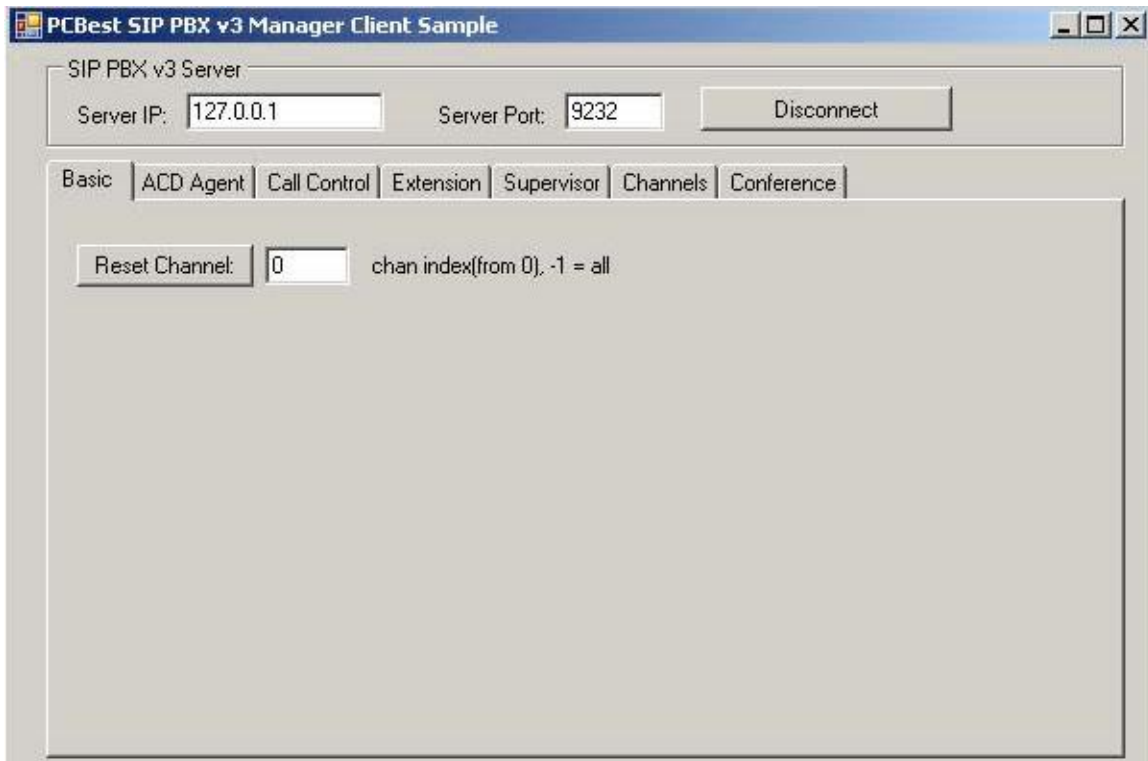
ServerConnected

This event is triggered when manage client connected to server or disconnect to server

Format: `void ServerConnected(bool bConnected)`
 bConnected: connected or not

Return: none

Channel related methods and events:



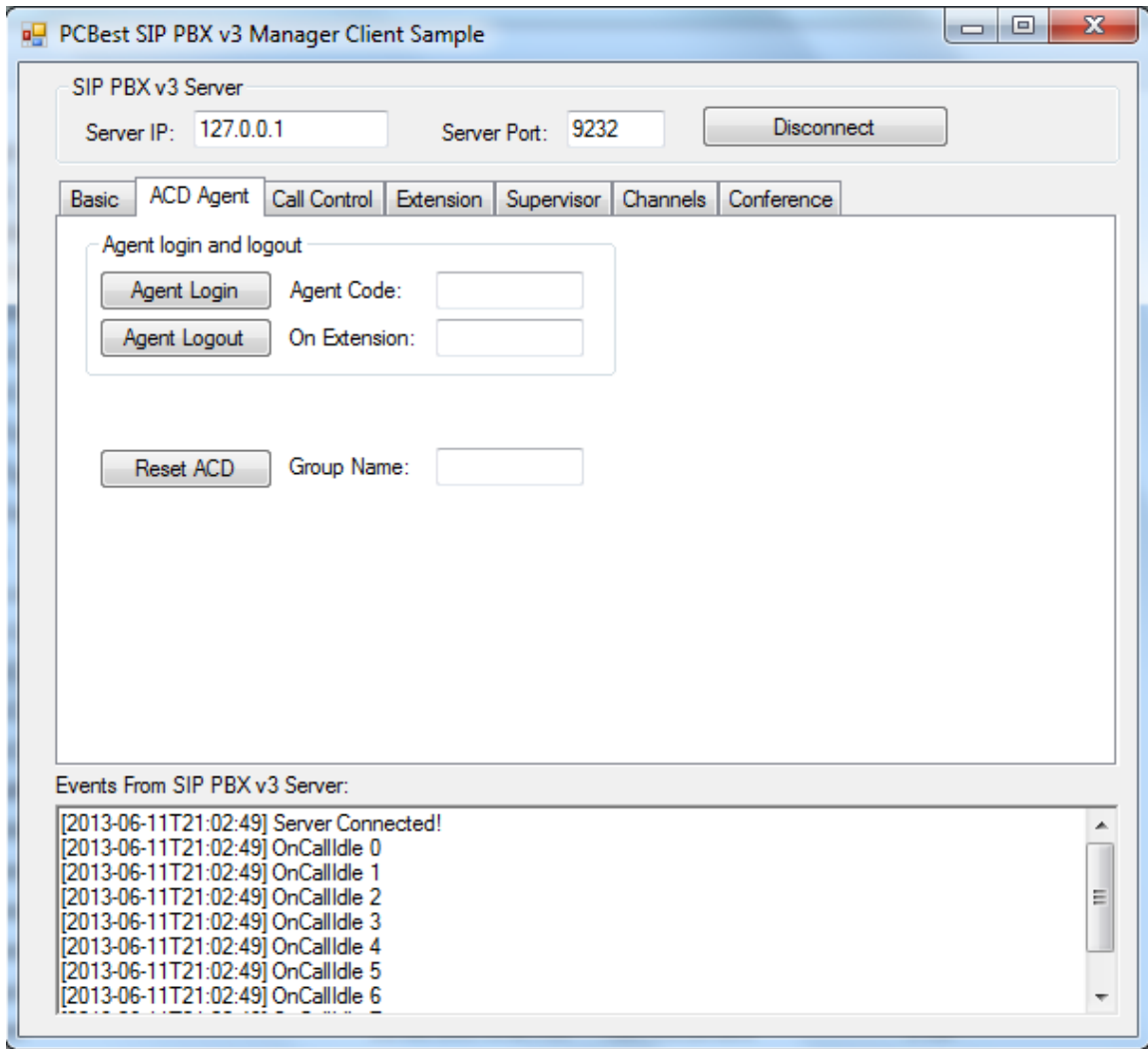
ResetChannel

Reset a channel. Disconnect a call on the channel if there is any.

Format: `void ResetChannel(int ch)`
 ch: the index of channel, based on 0.

Return: none, but it will trigger OnCallIdle event if there was a call on this channel

Agent related methods:



AgentLogin

Specify an agent login on an extension. It will trigger the event OnAgentLog.

Format: `public void AgentLogin(string agentCode, string extenNum, bool bLogin, string p1, string p2, string p3)`
 agentCode: the code of agent
 extenNum: the extension number
 bLogin: true=login false=logout
 p1,p2,p3: personal data for saving in database

Return: none, but it will trigger the event following.

OnAgentLog

The event when an agent login or logout.

Format: `public void OnAgentLog(bool bLogIn, string agentCode, string extenNum, string p1, string p2, string p3)`
 bLogIn: true=login false=logout
 agentCode: the code of agent

extenNum: the extension number
p1,p2,p3: personal data for saving in database

Return: none.

GetAgentStatus

Get the agent calling status. It will trigger the event `OnAgentStatus`.

Format: `void GetAgentStatus(string agentCode)`
agentCode: the code of agent

Return: none, it will trigger the event following.

OnAgentStatus

The event for agent status.

Format: `void OnAgentStatus(string agentCode, string atExten, string callStatus)`
agentCode: the code of agent
atExten: The extension number which agent is at (logged in).
callStatus: 0 = idle, 10 = offered, 20 = dialing, 21 = ringing, 30 = connected

Return: none

ResetACD

Reset a ACD group

Format: `void ResetACD(string acdName)`
acdName: the name of hunt group (ACD group).

Return: none

Call Control Related Methods and Events:



MakeCall

make a call out

Format: `string MakeCall(int ch, string caller, string callee)`
acdName: the name of hunt group (ACD group).
ch: the index of channel
caller: the caller in sip address format: `<sip:1234@abc.com>`.
callee: the called id in sip format: `<sip:456@def.com:5060>`.

Return: the command id for later on to get the result

MakeCall

make a call out

Format: `string MakeCall(int ch, string caller, string callee, string username, string passwd)`
ch: the index of channel
caller: the caller in sip address format: `<sip:1234@abc.com>`.
callee: the called id in sip format: `<sip:456@def.com:5060>`.
username: the user name for outbound call credential
passwd: the password for outbound call credential

Return: the command id for later on to get the result

MakeCall

make a call out

Format: `string MakeCall(int ch, string caller, string callee, string username, string passwd, string uri, string contact)`
ch: the index of channel

caller: the caller in sip address format: <sip:1234@abc.com>.
callee: the called id in sip format: <sip:456@def.com:5060>.
username: the user name for outbound call credential
passwd: the password for outbound call credential
uri: the request URI in SIP invite
contact: the contact address in SIP invite

Return: the command id for later on to get the result

AnswerCall

answer an incoming call on a channel

Format: `void AnswerCall(int ch)`
ch: the index of channel

Return: none, but it will trigger the event OnCallConnected if succeed.

HangupCall

disconnect call on a channel

Format: `void HangupCall(int ch)`
ch: the index of channel

Return: none, but it will trigger the event OnCallIdle if succeed.

HangupCall

disconnect call on a channel

Format: `void HangupCall(int ch, int reasonCode, string reasonDesc)`
ch: the index of channel
reasonCode: reason code
reasonDesc: reason description

Return: none, but it will trigger the event OnCallIdle if succeed.

HoldCall

hold call on a channel

Format: `void HoldCall(int ch)`
ch: the index of channel

Return: none, but it will trigger the event OnCallHold if succeed.

TransferCall

blind transfer call on a channel

Format: `void TransferCall(int ch, string callee) //blind transfer`
ch: the index of channel
callee: transferee sip address, like <sip:78646@pcbest.net>

Return: none

TransferCall

consult transfer call on a channel

Format: `void TransferCall(int ch, string callee, int ch1) //consult transfer`
ch: the index of channel
callee: transferee sip address, like <sip:78646@pcbest.net>
ch1: the index of another channel which is the address above but connected

Return: none

OnCallConnected

This event is triggered whenever there is a call connected

Format: `void OnCallConnected(int ch, string unique_id, string dialplan_name, string audio_fn)`
ch: the index of channel
unique_id: unique id to mark this call
dialplan_name: dialplan name will be used for this call
audio_fn: if recording, its file name.

Return: none

OnCallIdle

This event is triggered whenever a call got disconnected

Format: `void OnCallIdle(int ch, string unique_id, string dialplan_name, string audio_fn)`
ch: the index of channel
unique_id: unique id to mark this call
dialplan_name: dialplan name will be used for this call
audio_fn: if recording, its file name.

Return: none

OnCallRinging

This event is triggered whenever a outbound call is ringing(remote is ringing).

Format: `void OnCallRinging(int ch, string unique_id, string dialplan_name, string audio_fn)`
ch: the index of channel
unique_id: unique id to mark this call
dialplan_name: dialplan name will be used for this call
audio_fn: if recording, its file name.

Return: none

OnCallDialing

This event is triggered whenever a outbound call is dialing.

Format: `void OnCallDialing(int ch, string unique_id, string caller, string callee, string dialplan_name, string audio_fn)`

ch: the index of channel

unique_id: unique id to mark this call

caller: caller id

callee: callee id

dialplan_name: dialplan name will be used for this call

audio_fn: if recording, its file name.

Return: none

OnCallOffered

This event is triggered whenever there is a new incoming call

Format: `void OnCallOffered(int ch, string unique_id, string caller, string callee, string dialplan_name, string audio_fn)`

ch: the index of channel

unique_id: unique id to mark this call

caller: caller id

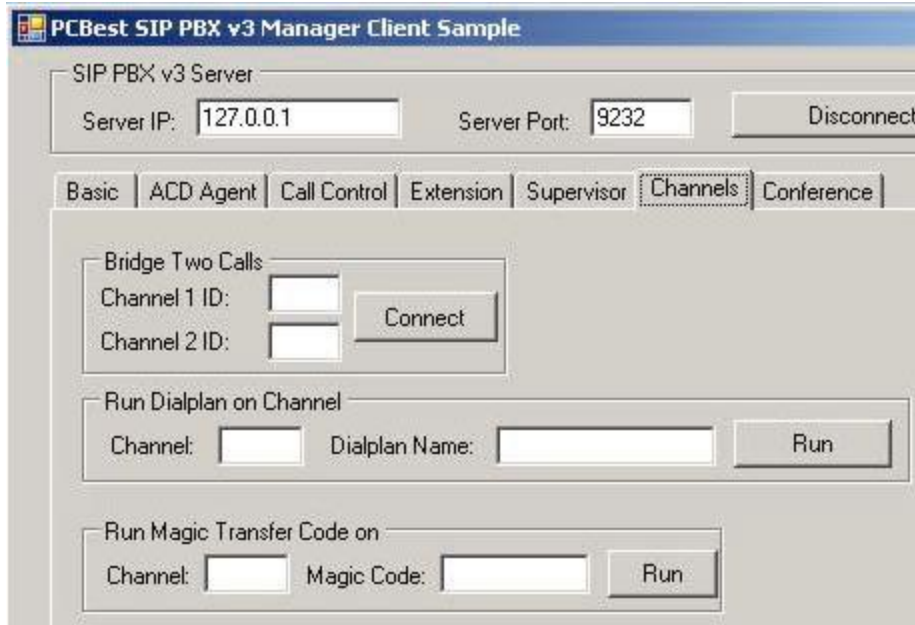
callee: callee id

dialplan_name: dialplan name will be used for this call

audio_fn: if recording, its file name.

Return: none

Channel related methods and events:



MagicTransfer

magic transfer call

Format: `void MagicTransfer(int ch, string transCode)`

ch: the index of channel

transCode: The magic transfer code

Return: none

BridgeTwoCalls

brige the calls on two channels

Format: `string BridgeTwoCalls(int ch1, int ch2)`

ch1: the index of channel 1

ch2: the index of channel 2

Return: the command id for later to get the command status

RunDialPlan

run a dialplan on the channel

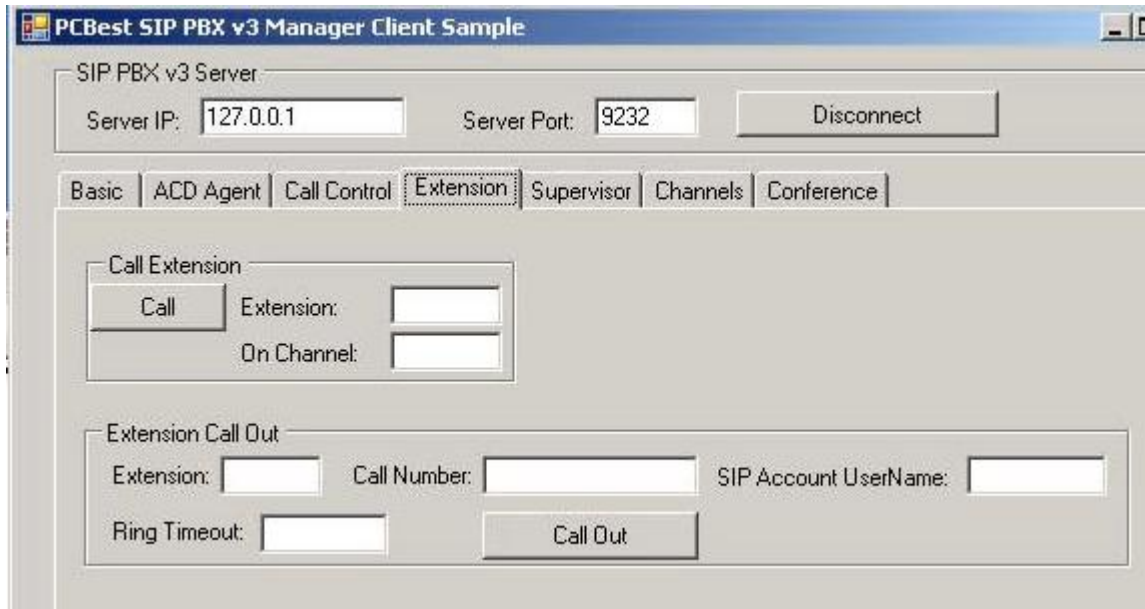
Format: `void RunDialPlan(int ch, string planName)`

ch: the index of channel

planName: the name of dialplan

Return: none

Extension related methods and events:



MakeExtensionCall

make a call to extension on specific channel

Format: `string MakeExtensionCall(int ch, string extnNum, string sCaller)`

ch: the index of channel

extnNum: the extension number

sCaller: caller id

Return: the command id

ExtenCallOut

Initiate a call from an extension to outside. It actually uses auto-dialer task to dial out then connect with extension once the call is connected.

Format: `void ExtenCallOut(string extnNum, string destNum, string`

`sipAcctUserName, int ringTimeoutSec)`

extnNum: the extension number

destNum: the destination number

sipAcctUserName: the sip account name to be used for outbound call

ringTimeoutSec: how many seconds to wait in the ring

Return: none

ExtenCallOutEx

Initiate a call from an extension to outside. It actually uses auto-dialer task to dial out then connect with extension once the call is connected.

Format: `void ExtenCallOutEx(string extnNum, string destNum, string sipAcctUserName, int ringTimeoutSec, bool enableDetect, bool disconnectAfterDetect)`

extnNum: the extension number

destNum: the destination number

sipAcctUserName: the sip account name to be used for outbound call

ringTimeoutSec: how many seconds to wait in the ring

enableDetect: if enable human/answering machine detection

disconnectAfterDetect: if disconnect call after detection is done.

Return: none

OnExtenStatus

The event to reflect extension status

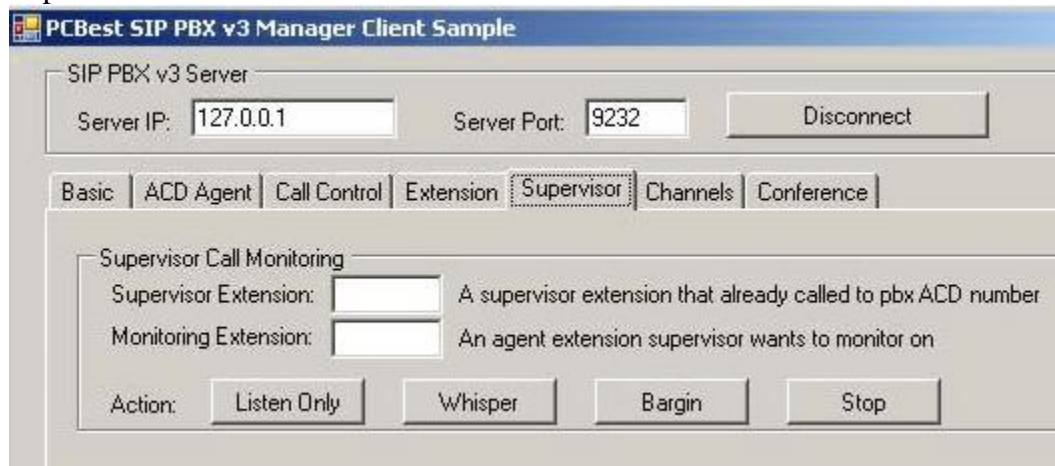
Format: `void OnExtenStatus(string extenNum, string callStatus)`

extenNum: the extension number

callStatus: 0 = idle, 10 = offered, 20 = dialing, 21 = ringing, 30 = connected

Return: none

Supervisor feature to monitor extension's call.



MonitorCall

Connect supervisor's extension with agent/user extension to allow supervisor monitor the current calls.

Format: `void MonitorCall(string extnSupervisor, string extnNormal, int monitorType)`

extnSupervisor: the supervisor extension

extnNormal: the extension number to be monitored.

monitorType: 0 = listen, 1 = whisper, 2 = talking(bargin), -1 = stop

monitoring(get out, withdraw)

Return: none

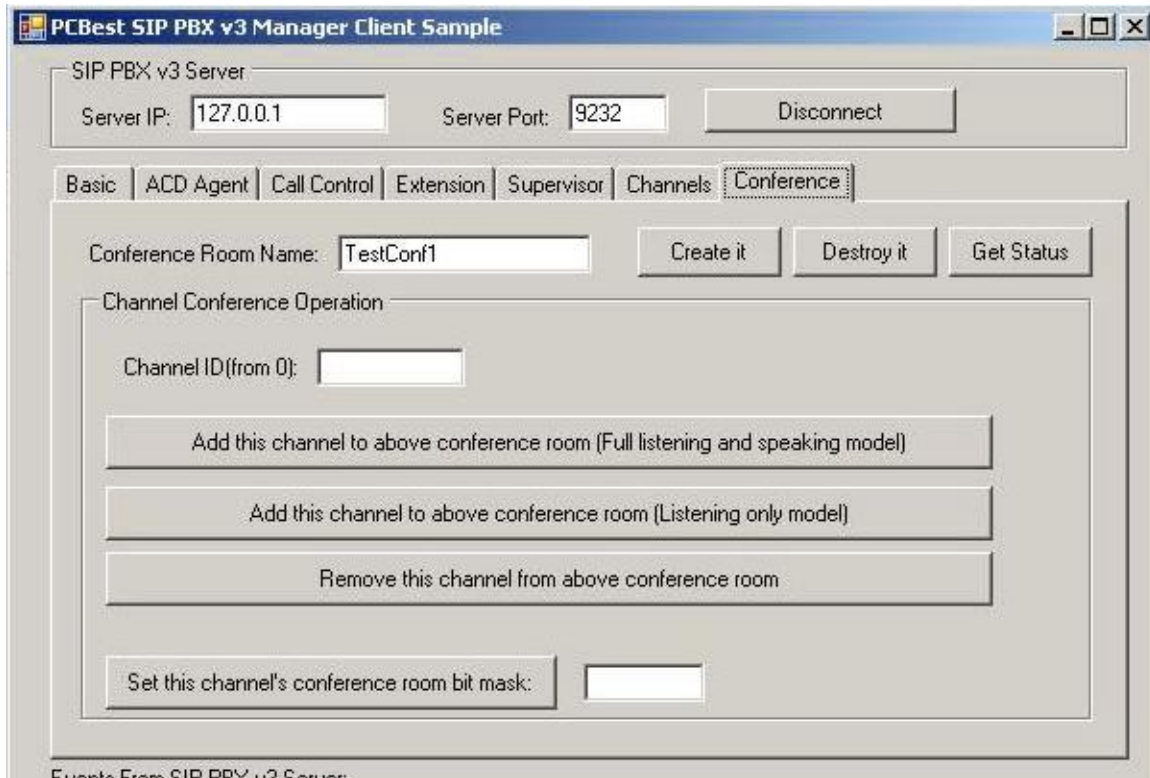
OnCallMonitoring

Monitoring call event

Format: `void OnCallMonitoring(string extenSupervisor, string extenNormal, int monitorType)`
 extnSupervisor: the supervisor extension
 extnNormal: the extension number to be monitored.
 monitorType: 0 = listen, 1 = whisper, 2 = talking(bargin), -1 = stop
 monitoring(get out, withdraw)

Return: none

Conference related methods and events



CreateConferenceRoom

Create a conference room on PBX dynamically.

Format: `void CreateConferenceRoom(string conf_name)`
 conf_name: the name of conference room

Return: none

DestroyConferenceRoom

Destroy a conference room on PBX dynamically.

Format: `void DestroyConferenceRoom(string conf_name)`
conf_name: the name of conference room

Return: none

SetChanInConferenceRoom

This function is majorly used to send a channel into a conference room, or withdraw it.

Format: `void SetChanInConferenceRoom(int ch, string conf_name, int opt)`
ch: the index of the channel
conf_name: the name of conference room
opt: 0 = take out of conference room. 1 = add into conference room. 2 = monitor(listening only, not speaking)

Return: none

SetChanConferenceBitMask

Set channel's bitmask in conference room. Set channel's output when in conference room.

This function is used to disable the chan's output voice to other channels in the same conference.

Default channel mask is always 0xFFFFFFFF, which means output to all other channels in the conference room.

Every bit marks a channel. If the bit is 1, its voice can output to the channel.

The First channel in the conference room is 0x01.

The second channel in the conference room is 0x02.

The third channel in the conference room is 0x04.

So if you want the channel's output goes to the first channel, and the third channel, you can set this for this channel:

`SetChanConfMaskch, 0x05); //which 0x05 = 0x01 + 0x04`

Another example,

1st channel is connected with Agent. (Channel Index is 0, and it is the first channel set to the conference room)

2nd channel is connected with Customer. (Channel Index is 1, and it is the second channel set to the conference room)

3rd channel is supervisor. (Channel Index is 2, and it is the third channel set to the conference room)

They are all in the same conference room. Regularly if don't set anything, they can hear each other.

If supervisor only wants the agent hear his voice, not the customer, you can do so:

`SetChanConfMask(2, 0x01);`

It means that only the first channel get his voice.

Format: `void SetChanConferenceBitMask(int ch, uint bitMask)`
ch: the index of the channel
bitMask: bit mask to enable or disable output

Return: none

GetConferenceRoomStatus

Trigger the conference room event to get the status

Format: `void GetConferenceRoomStatus(string conf_name)`
conf_name: the name of conference room

Return: none, but the event OnConferenceRoomStatus will triggered.

OnConferenceRoomStatus

The event to receive current conference status

Format: `void OnConferenceRoomStatus(string roomName, string channels)`
roomName: the name of conference room
channels: channel status in the conference room. the format is:
channel,status;channel,status;channel,status
status: 1 = listen and speak, 2 = listening only(monitoring)

Return: none.

Please refer to the source code of manager client about full demonstration. The demo source code is in C#, and if you are .NET developer, you can easily use it in your project. It provides very simple interfaces to use. But if you are like vb6, Delphi developer, and you want develop manager client application in your own language, here is guide how to do:

Assume you can use vb6 to open a TCP connection to IPPBXv3's manager port(you can set this in ippbxv3's GUI, default it is 9232). After connected, you will receive events like this:

command parameter1|parameter2|parameter3.....

For new incoming call, you will receive command CallOffered. Format like this:
CallOffered channel-id|unique-id|caller|callee|dialplan|recording-audio-filename

For call dialing out, you will receive command like this:
CallDialing channel-id|unique-id|caller|callee|dialplan|recording-audio-filename

If remote ringed for outbound call, you will receive:

CallRinging channel-id|unique-id|dialplan|recording-audio-filename

If call got connected, the event looks like:

CallConnected channel-id|unique-id|dialplan|recording-audio-filename

If call got disconnected, the command format is:

CallIdle channel-id|unique-id|dialplan|recording-audio-filename

There are other commands, and if you need, please contact PCBest Networks support for more details.

7.3 Database Development (V3)

PBX v3 is a completely database driven engine. It saves everything into database table. For example, real-time status of PBX are saved into status_xxx.

Tables:

cdr_xxx are CDR tables.

auto_dialer_xxx are auto dialer tables.

cfg_xxx are PBX configuration tables.

If you want to develop your own user interfaces, like web interface, to work with PBX, **cfg_xxx** tables are the tables you mostly need to deal with. Each **cfg** table has a field **ModTag**, which makes this record's status.

If you add or change a record, you need to set ModTag to 1. PBX service will later refresh its memory and set this tag back to 0.

If you want to remove(delete) the record, you need to set ModTag to 2. PBX service will later delete it from table.

When ModTag is 0, then it means there is no change on this record.

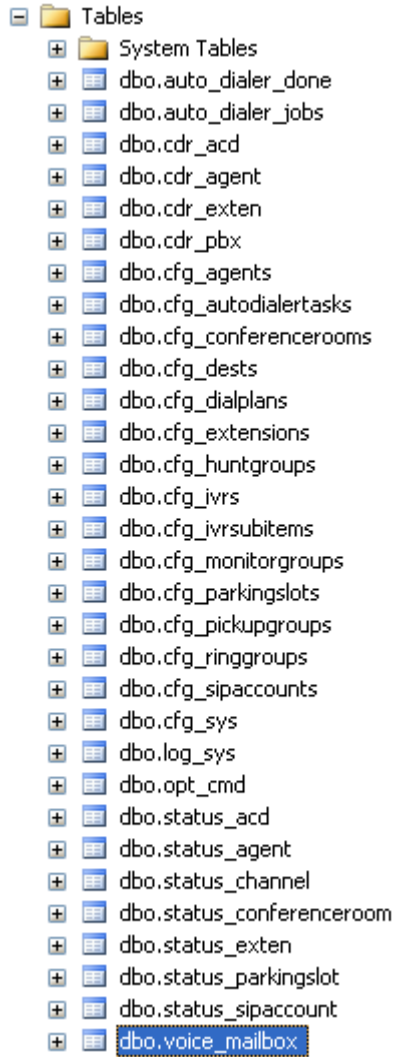
log_xxx are PBX real-time log table.

opt_cmd are PBX command table. PBX checks this table regularly to see if there are commands sent to PBX through DB.

status_xxx are PBX real-time status table.

voice_mailbox is voice mailbox table.

Here is the full list of database table of PBX v3:

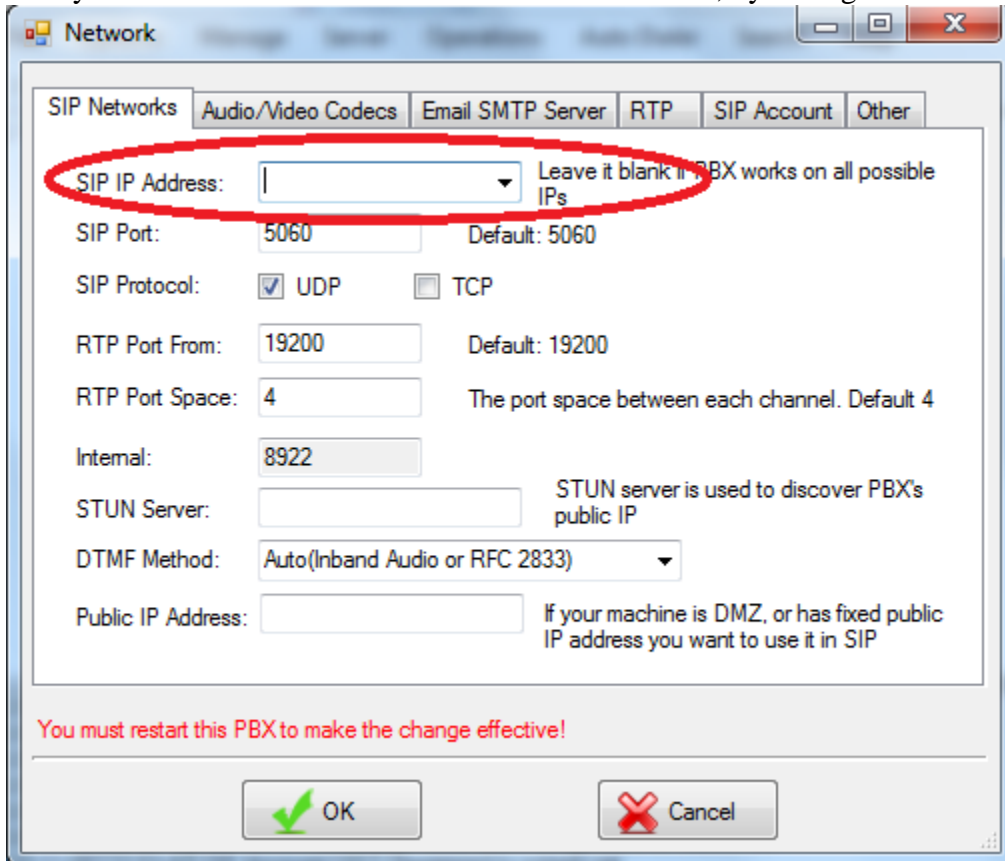


For more detail info about database development of PBX v3, please contact PCBest at support@pcbest.net

8 Session Border Controller (SBC)

Sometimes you have need to transfer calls between LAN and WAN. PCBest SIP PBX has flexible features to allow you do so.

First you will need to listen on all NICs for SIP address, by setting it to blank:



Then we can set up the individual cases.

WAN to LAN

Create a SIP account, named Call-LAN, which is used to call out to LAN, by marking it as SIP trunk (assuming 192.168.1.10 is the IP address of another internal server to handle calls):

Add SIP Account

Basic | DID's | Peer2Peer

☒ It is a SIP trunk

Display Name: Sample: Bob Wall, Company1, Trunk1

User Name: Sample: 7184773245, 1001, or Mike

SIP Domain: Sample: pcbest.net, voip.com

SIP Proxy: Sample: pcbest.net, usually same as domain

SIP Protocol: ☒ UDP ☐ TCP

Authorization: Sample: 7845, usually same as UserName

Password: Your secret code

Expire Duration: In seconds, default is 3600 = 1 hour

☐ Register with SIP proxy server to receive incoming calls

OK Cancel

Then create a outbound dialplan, named Call-LAN, to use this SIP account(Call-LAN) :

Dian Plan

Basic | Time Schedule | Extensions or Agents

Plan Name: Any name you like to give for this plan

Call Direction: ☐ Inbound ☒ Outbound Which call direction the plan is for

Caller Number: Blank if no limit on caller. Use * for any number, and ? for any one digit. You can use @ for calls on specific IP/domain in SIP format. For example: *@192.168.0.2

Called Number:

Plan Template:

Pre-strip: Outbound called number pre-strip text
For example: prestrip text for called number 9* is 9.

Pre-append: Pre-append string after pre-strip.

Use SIP Account: Which SIP account you want to use for outbound call

Alter SIP Account: Second SIP account in case the first one is offline

OK Cancel

Then create an inbound dialplan, named WAN-IN, set called number as *@76.39.134.65
Assume your public IP address is 76.39.134.65:

The screenshot shows the 'Dian Plan' configuration window with the following fields and values:

- Plan Name:** WAN-IN
- Call Direction:** Inbound (selected)
- Caller Number:** (empty)
- Called Number:** *@76.39.134.65
- Plan Template:** Call Forward
- Outbound Plan:** Call-LAN
- Pre-strip:** (empty)
- Pre-append:** (empty)
- Use SIP Account:** (empty)
- Alter SIP Account:** (empty)

Buttons at the bottom: Finish (green checkmark icon), Cancel (red X icon).

This dialplan will handle all calls to 76.39.134.65, and forward it to outbound plan Call-LAN.

LAN to WAN

You can also follow above steps to create a route for LAN to WAN.