Call Center Manager

Table of contents

[Introduction 2](#_Toc354727433)

[Software requirements 2](#_Toc354727434)

[Project development 2](#_Toc354727435)

[Implementation 5](#_Toc354727436)

[Model 5](#_Toc354727437)

[RealClient 5](#_Toc354727438)

[PhoneNumberStatistics 6](#_Toc354727439)

[UserStatistics 6](#_Toc354727440)

[UserStatisticsContainer 6](#_Toc354727441)

[Statistics 6](#_Toc354727442)

[View 6](#_Toc354727443)

[AboutBox 6](#_Toc354727444)

[BaseWindow 6](#_Toc354727445)

[ConnectToServerWindow 6](#_Toc354727446)

[MainWindow 7](#_Toc354727447)

[Presenter 7](#_Toc354727448)

[ConnectToServerPresenter 7](#_Toc354727449)

[MainWindowPresenter 8](#_Toc354727450)

[Util 9](#_Toc354727451)

[SimpleIOCContainer 9](#_Toc354727452)

# Introduction

The basic task of the Call Center Manger sample program, connected to Ozeki Phone System XE, is to display statistics and to implement certain call features in the system. Such feature is call interception and call merging. The purpose of this documentation is to show step-by-step how an application like this can be developed. The system will be developed under Windows Operating System and with the help of Visual Studio.

# Software requirements

Microsoft Windows

Microsoft Visual Studio

Ozeki Phone System XE

# Installation and Configuration of Ozeki Phone System XE

In order to connect to the server, Ozeki Phone System XE has to be installed. A Setup Guide can be found here: <http://www.ozekiphone.com/how-to-install-and-configurate-your-ozeki-phone-system-xe-172.html>, the installation package can be downloaded from the following page: <http://www.ozekiphone.com/download-21.html>. After the installation, the main page of the PBX can be accessed on the <http://localhost:7777/Home> address.

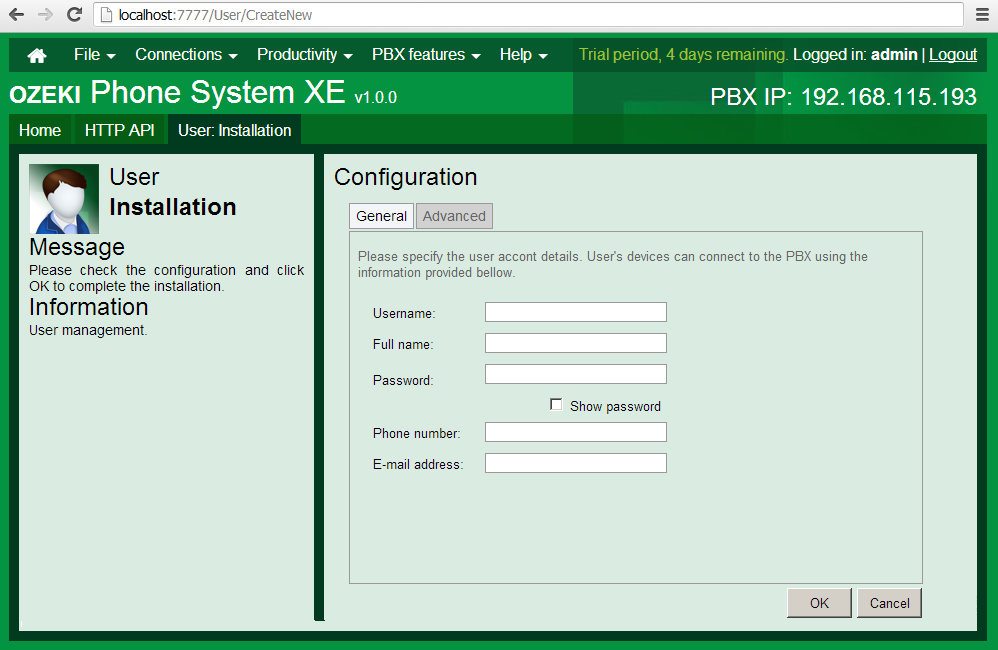


For the first login attempt, use the username and password, which was provided during the installation process.

New user can be added to the system with the Add user button. It can be found under the Office users menu item in the Connections menu.



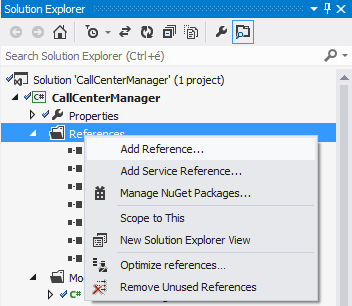
The username and password specified here can be used to connect to the system, in the **OPSClient LoginAsync** method.



# Project development

1. Let us create a new Windows Forms Application project with the help of File/New/Project menu item.

2. Let us add the OPSSDK.dll file to the project references in the Solution Explorer by clicking on Add Reference… menu item. This will ensure the connection interface to Ozeki Phone System XE.



# Implementation

During the development of the application, we followed the Model-View-Presenter software designing pattern. This is useful because the display interface can be easily changed.

## Model

The files that include the application logics were added into the Model namespace.

### RealClient

With the help of the **Login** method, we can connect to Ozeki Phone System XE and try to login with the default username and password in an asynchronous way.

public void Login(string server\_address, string username, string password)

{

stats = new Statistics();

ops\_client = new OpsClient();

ops\_client.ErrorOccured += OPSClientOnErrorOccured;

ops\_client.LoginAsync(server\_address, username, password, Completed);

}

In case of successful connection, we subscribe to the below events of the **OpsClient**:

ops\_client.SessionCreated += OPSClientOnSessionCreated;

ops\_client.SessionCompleted += OPSClientOnSessionCompleted;

ops\_client.PhoneBookChanged += OPSClientOnPhoneBookChanged;

In case of unsuccessful connection attempt the taskmanager of the **OpsClient ErrorOccured** runs.

The **SessionCreated** event signals if there is a new incoming call in the system. The status changes of the new call can be handled in the **SessionStateChanged** event. In our sample program, we monitor the statistics of the dropped calls and activate an event with the new statistic data, in case a call is dropped by the system.

if (args.Item == SessionState.CalleeUnavailable)

{

lock (sync)

{

stats.NumberOfDroppedSessions++;

}

var stats\_handler = StatisticsChanged;

if (stats\_handler != null) stats\_handler(this,

new VoIPEventArgs<Statistics>(stats));

}

When a call ends, the event of the **OpsClientSessionCompleted** runs. Here, the statistics is modified according to the call duration, whether it was less than a minute or not. The **RealClientSessionComleted** is also activated, on which the presenter will subscribe.

### PhoneNumberStatistics

In this class statistics are stored, which belong to one phone number. The sample program will only display the **CallsInProgress** property but the number of all calls, including the outgoing and incoming ones, will be stored.

### UserStatistics

The **PhoneBookItem** includes a contact and the phone number statistics, which belong to it.

### UserStatisticsContainer

**UserStatisticsContainer** creates a statistics summary from the phone book that was returned by the **OpsClient.**

### Statistics

This class will store the statistics data, here every feature that is significant to us can be stored.

## View

The user interfaces and the unique controls have been included in this namespace. From the unique controls, the **FlickerFreeTreeView** has to be introduced because otherwise the display would flicker in the treeview. The **VerticalProgressBar** could be useful for displaying the statistics in percentage but it is not used in this sample.

### AboutBox

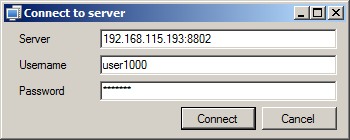
This is the about of the application.

### BaseWindow

BaseWindow is the parent of **ConnectToServerWindow** and **MainWindow**. This interface implements basic features like error message, informationmessage display, window closing or waiting.

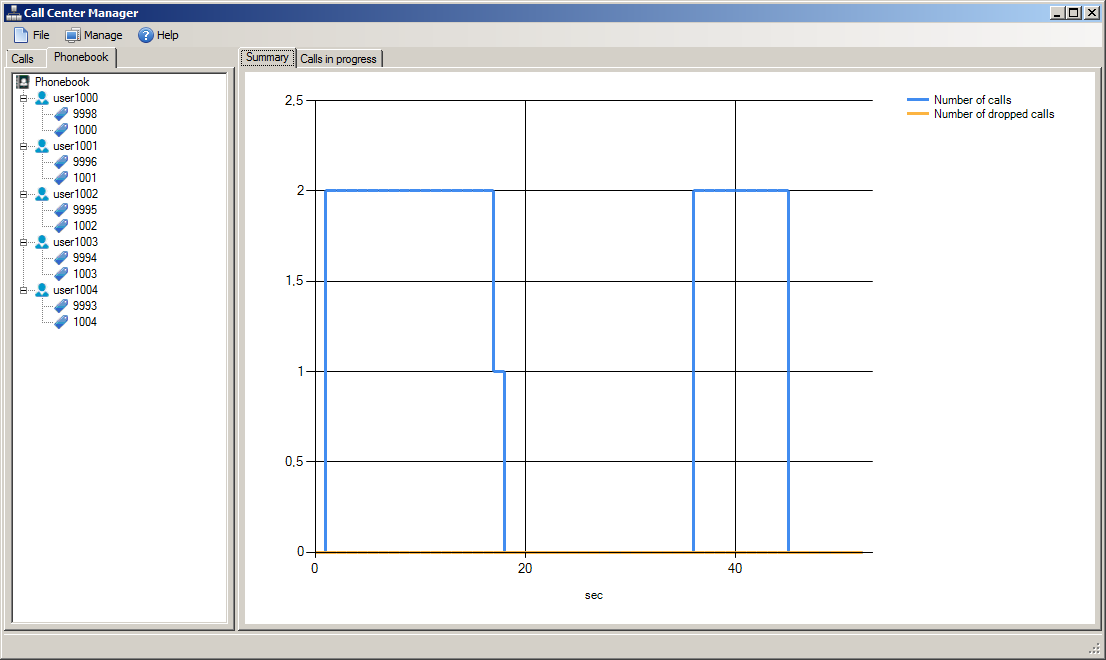
### ConnectToServerWindow

This windows makes it possible to connect to Ozeki Phone System XE. In order to connect, the followings have to be provided: the IP address of the computer (ex.: 192.168.115.193:8802) on which Ozeki Phone System XE runs, username (Ozeki Phone System office user), password. When starting the Call Center Manager, this window appears automatically but the opportunity is given to connect to a provider later on.



### MainWindow

The main window of the application. Here, on the Calls tab, the system calls can be seen in a tree structure. On the Phonebook tab, there are the contacts with the phone numbers, which belong to them. This interface displays the statistics data and gives the opportunity to export the phone book in csv file format. Here we can disconnect from the server and connect to another one. By clicking on the selected call, in the Calls treeview, with the right mouse button in the context menu, we have the chance to listen or speak into a call. The two buttons above the Calls treeview also serve this purpose.



## Presenter

These files are responsible for connecting the model and the view.

### ConnectToServerPresenter

One dependency of the ConnectToServerPresenter is an **IConnectToServer** interface, which is implemented by the **ConnectToServerWindow**. We connect to the server with this view. The other dependency is the **IOPSClient**, which is implemented by the **RealClient** class. The connection is done via the client interface (**IOPSClient**) and also the user interface is notified about the connection process. In case of unsuccessful connection attempt an error message appears, anyway we close the GUI.

### MainWindowPresenter

The **MainWindowPresenter** also has two dependencies, from which the **IOPSClient** is completely the same as the object being used by the **ConnectToServerPresenter**. Since, the IOPSClient is added to the IoC container as a singleton class. We will talk about the IoC container later on. The other dependency is the **IMainWindow** interface, which is implemented by the **MainWindow**. Here, the **IMainWindow** interface also notifies the the User Interface if any change needs to be done in the display.

In this presenter, the default micropohone is instantiated.

readonly Microphone microphone;

microphone = Microphone.GetDefaultDevice();

with a default speaker.

readonly Speaker speaker;

speaker = Speaker.GetDefaultDevice();

Both devices are the descendants of the **AudioHandler**, so they can be connected to the call with the **ISession ConnectAudioSender** and **ConnectAudioReceiver** methods.The **Speaker** is connected with the **ConnectAudioSender** method and the **Microphone** is connected with the **ConnectAudioReceiver** method. There are more classes that are the descendants of the **AudioHandler**. Each of these classes can be connected to the **ISession**. We have the opportunity to play audio files e.g. with the help of the **MP3StreamPlayback** class or by using text to speech feature with the **TextToSpeech** class. For more information about the AudioHandler class, visit the following page: <http://voip-sip-sdk.com/doc>, Ozeki VoIP SIP SDK included with the sample programs, can be downloaded from: <http://voip-sip-sdk.com/>

The following presenter will subscribe to the main events of **IOPSClient**. In the sample program we subscribed to the events below:

client.ErrorOccurred += ClientOnErrorOccurred;

client.SessionCreated += ClientOnSessionCreated;

client.SessionCompleted += ClientOnSessionCompleted;

client.PhoneBookChanged += ClientOnPhoneBookChanged;

client.StatisticsChanged += ClientOnStatisticsChanged;

If a call arrives to the system, it will appear both in the Calls and Phonebook treeviews. In order to display them, we need to subscribe to the **SessionCreated**, **SessionCompleted** events.

## Util

In this namespace, the dialogue windows that are responsible for displaying the messages, constants, delegates and the IoC container can be found.

### SimpleIOCContainer

The IoC (inversion of control) container is a singleton class, which is suitable for storing dependencies:

SimpleIOCContainer.Instance.AddDependency<IOPSClient>(() => new RealClient());

The following code is for dependency resolving:

SimpleIOCContainer.Instance.Resolve<IOPSClient>();

It makes dependency replacement available at the same place so changing to the usage of test classes will be easier.