```
using Ozeki.Network.Nat;
using Ozeki.VoIP;
using Ozeki.VoIP.SDK;
ISoftPhone softphone;   // softphone object
IPhoneLine phoneLine;   // phoneline object
softphone = SoftPhoneFactory.CreateSoftPhone(5000, 10000, 5060);
softphone.IncomingCall += softphone_IncomingCall;
public void Register(bool registrationRequired, string displayName, string userName, string
authenticationId, string registerPassword, string domainHost, int domainPort)
{
   try
   {
      var account = new SIPAccount(registrationRequired, displayName, userName,
authenticationId, registerPassword, domainHost, domainPort);
      Console.WriteLine("\n Creating SIP account {0}", account);
      var natConfiguration = new NatConfiguration(NatTraversalMethod.None);

      phoneLine = softphone.CreatePhoneLine(account, natConfiguration);
      Console.WriteLine("Phoneline created.");

      phoneLine.PhoneLineStateChanged += phoneLine_PhoneLineStateChanged;

      softphone.RegisterPhoneLine(phoneLine);
   }
   catch(Exception ex)
   {
      Console.WriteLine("Error during SIP registration" + ex.ToString());
   }
}
ICall call;
MediaConnector mediaConnector;
AudioHandler audioHandler;
PhoneCallAudioSender phoneCallAudioSender;
Timer greetingMessageTimer;
public CallHandler(ICall call)
{
     greetingMessageTimer = new Timer();
     greetingMessageTimer.Interval = 30000;
     greetingMessageTimer.Elapsed += greetingMessageTimer_Elapsed;
     this.call = call;
     phoneCallAudioSender = new PhoneCallAudioSender();
     phoneCallAudioSender.AttachToCall(call);
     mediaConnector = new MediaConnector();
}
```

```
public void Start()
{
   call.CallStateChanged += call_CallStateChanged;
   call.DtmfReceived += call_DtmfReceived;
   call.Accept();
}
void call_DtmfReceived(object sender, VoIPEventArgs<DtmfInfo> e)
{
   DisposeCurrentHandler();
   switch (e.Item.Signal.Signal)
   {
       case 0: break;
       case 1: TextToSpeech("Product XY has been designed for those software developers who
especially interested in VoIP developments. If you prefer .NET programming languages, you
might be interested in Product XY."); break;
       case 2: MP3ToSpeaker(); break;
   }
}
private void TextToSpeech(string text)
{
   var tts = new TextToSpeech();
   audioHandler = tts;

   mediaConnector.Connect(audioHandler, phoneCallAudioSender);
   tts.AddAndStartText(text);
}
private void MP3ToSpeaker()
{
   var mp3Player = new MP3StreamPlayback("../../test.mp3");
   audioHandler = mp3Player;

   mediaConnector.Connect(mp3Player, phoneCallAudioSender);
   mp3Player.StartStreaming();
}
static void Main(string[] args)
{
     callHandlers = new List<CallHandler>();
     var softphone = new Softphone();

     Console.WriteLine("/* Program usage description */");

     sipAccountInitialization(softphone);

     softphone.IncomigCall += softphone_IncomigCall;
```

```
     Console.ReadLine();
}
private static void sipAccountInitialization(Softphone softphone)
{
    Console.WriteLine("Please setup your SIP account!\n");
    Console.WriteLine("Please set your authentication ID: ");
    var authenticationId = Read("authenticationId", true);

    Console.WriteLine("Please set your user name (default:" +authenticationId +"): ");
    var userName = Read("userName", false);
    if (string.IsNullOrEmpty(userName))
        userName = authenticationId;

    Console.WriteLine("Please set your name to be displayed (default: " +authenticationId +"): ");
    var displayName = Read("displayName", false);
    if (string.IsNullOrEmpty(displayName))
        displayName = authenticationId;

    Console.WriteLine("Please set your registration password: ");
    var registrationPassword = Read("registrationPassword", true);

    Console.WriteLine("Please set the domain name (default: your local host): ");
    var domainHost = Read("domainHost", false);
    if (string.IsNullOrEmpty(domainHost))
        domainHost = NetworkAddressHelper.GetLocalIP().ToString();
    Console.WriteLine(domainHost);

    Console.WriteLine("Please set the port number (default: 5060): ");
    int domainPort;
    string port = Read("domainPort", false);
    if (string.IsNullOrEmpty(port))
    {
        domainPort = 5060;
    }
    else
    {
        domainPort = Int32.Parse(port);
    }
    Console.WriteLine("\nCreating SIP account and trying to register...\n");
    softphone.Register(true, displayName, userName, authenticationId, registrationPassword,
domainHost, domainPort);
}
private static string Read(string inputName, bool readWhileEmpty)
{
```

```
    while (true)
    {
        string input = Console.ReadLine();

        if (!readWhileEmpty)
        {
            return input;
        }

        if (!string.IsNullOrEmpty(input))
        {
            return input;
        }

        Console.WriteLine(inputName +" cannot be empty!");
        Console.WriteLine(inputName +": ");
   }
}
static void softphone_IncomigCall(object sender,
Ozeki.VoIP.VoIPEventArgs<Ozeki.VoIP.IPhoneCall> e)
{
    Console.WriteLine("Incoming call!");
    var callHandler = new CallHandler(e.Item);
    callHandler.Completed += callHandler_Completed;

    lock (callHandlers)
         callHandlers.Add(callHandler);

    callHandler.Start();
}
Console.WriteLine("Please set the number for blind transferring! If you don't want to use this
function of the IVR, press 0!");
blindTransfer = Read("blindTransfer", true);
public void BlindTransferNumber(string blindTransfer)
{
         blindTransferNumber = blindTransfer;
}
case 3:
                {
                   if (blindTransferNumber == "0")
                   {
                      TextToSpeech("You did not add any number for blind transferring!");
                      break;
                   }
```

```
                    else
                    {
                        call.BlindTransfer(blindTransferNumber);
                        break;
                    }
                }
PhoneCallAudioReceiver phoneCallAudioReceiver;
IEnumerable<string> choices;
SpeechToText stt;
public CallHandler(ICall call)
{
    greetingMessageTimer = new Timer();
    greetingMessageTimer.Interval = 30000;
    greetingMessageTimer.Elapsed += greetingMessageTimer_Elapsed;
    this.call = call;
    phoneCallAudioSender = new PhoneCallAudioSender();
    phoneCallAudioSender.AttachToCall(call);
    mediaConnector = new MediaConnector();
    phoneCallAudioReceiver = new PhoneCallAudioReceiver();
    phoneCallAudioReceiver.AttachToCall(call);
    choices = new List<string>() { "first", "second"};
    stt = SpeechToText.CreateInstance(choices);
}
public void Start()
{
   mediaConnector.Connect(phoneCallAudioReceiver, stt);
   call.CallStateChanged += call_CallStateChanged;
   stt.WordHypothesized += CallHandler_WordHypothesized;
   call.Accept();
}
void CallHandler_WordHypothesized(object sender, SpeechDetectionEventArgs e)
{
    DisposeCurrentHandler();
    Console.WriteLine(e.Word.ToString());
    switch (e.Word.ToString())
    {
        case "first": TextToSpeech("Product XY has been designed for those software developers
who especially interested in VoIP developments. If you prefer .NET programming languages,
you might be interested in Product XY."); break;
        case "second": MP3ToSpeaker(); break;
    }
}
```