OPS Agent Dialer

# Bevezetés

The OPS Agent Dialer lehetővé teszi, hogy az Ozeki Phone System-hez csatlakozva kifelé indítsunk hívásokat egy megadott lista alapján. Az Agent Dialer prediktív módon képes átirányítani a létrejött hívásokat a szabad agent-ek egyikére, tehát csak abban az esetben irányítja a hívásokat az ügynökökhöz, hogyha a hívás sikeres volt. Egymás mellett több hívás is elindulhat a szabad ügynökök számától függően. Ezáltal időt takarítva meg az ügynökök számára, és ez növeli a produktivitást. Azügynökök kiválasztása többféle algoritmussal történhet.

# Szoftverkövetelmények

Microsoft Windows 7, 8, Vista

Microsoft Visual Studio 2012, .NET 4.0

# A példaprogram használata

Induláskor megjelenik a login képernyő, a hol meg kell adni az Ozeki Phone System SDK szolgáltatásának címét és portját, az Ozeki Phone System-ben az office users menüpontjában létrehozott felhasználó nevét és jelszavát.

## Beállítások

Bejelentkezés után a settings menüpontra kattintva szükség az alap működési beállításokat megadni. Az Agents résznél ki kell választani, hogy milyen Ozeki Phone System felhasználók lesznek az ügynökök, akikhez az ügyfélhívások menni fognak. A retry settingsben megadható, hogy a hívások újbóli elindítása után mely hívásállapotú ügyfeleket kísérelje meg újra felhívni a rendszer (pl. visszautasított hívás). A concurrent calls érték megadja, hogy egy időben hány párhuzamos hívás futhat. Ez az érték függ az ügynökök számától is, tehát legfeljebb annyi párhuzamos hívás lehet, ahány agent van a rendszerben beállítva. Az agent selection-nél az ügyfélhívások elosztásának módja adható meg.

Az API extension beállításához az Ozeki Phone System-ben fel kell venni egy API extension-t (Extensions/Install new), amelyet ez a példaprogram fog használni, majd ennek az extensionnek az azonosítóját kell itt megadni.

## Működés

Az open ikon-ra kattintva megnyitható egy ügyfél lista csv fájl, amiben nevek és telefonszámok vesszővel elválasztva szerepelnek (lásd ExampleCsv.csv fájl) . Betöltés után a Start ikonra kattintva indul el az ügyfelek hívása a lista alapján. Működés közben látszik a hívásoknak az állapotváltozása (in progress, completed, rejected stb.), a sikeres hívások áttranszferálódnak az aktuálisan szabad ügynökökhöz.

Ha végetértek a hívások, a start gomb újra aktívvá válik, ekkor lehetőség van a hívási folyamatot újra elindítani. Az idle állapotú még hátralévő, illetve a beállításokban beállított állapotú (rejected, not found stb.) úgyfeleket sorba elkezdi hívni a program.

# Implementáció

Az implementáció a projektben hivatkozza felhasználja a OzCommon és az OzCommonBroadcasts projekteket, arra épülnek az Agent Dialer GUI elemei. További funkciók is vannak, melyeket a közös projektekből használ a példaprogram.

## Model

Az AgentDialerViewModel model osztályban példányosítódik az AgentDialer osztály, ebben lesznek majd ütemezve a hívások, illetve végrehajtja a hívás átirányításokat és kezeli az ügynököket. Az AgentDialer megkapja azokat a beállításokat, amelyek az IGenericSettingsRepository–ban tárolva vannak. Beállításokat tároló model az AppPreferences osztály, melyben le vannak tárolva a jelenleg beállított ügynökök (AgentInfo). Ez alapvetően egy felhasználót (PhoneBookInfo) reprezentál az Ozeki Phone System-ben, de emellett működés közben az AgentEntry osztály tartalmazza többek közt az ügynök állapotát, eddigi hívásainak számát stb. A beállítások között szerepelnek az újrapróbálandó állapotok nevei, a maximális egyidejű hívások száma (ami nem lehet nagyobb, mint az ügynökök száma), az ügynök kiválasztási módszere és a beállított API extension azonosító. Az InitSettings() metódusban az alapértelmezett értékek állítódnak be, az OpsClient-től lekéri a rendszer a phone book tartalmát, hogy a felhasználók közül ki lehessen választani az ügynököket a settings ablakban.

Mivel az AgentDialerViewModel a BroadCastMainViewModel<T>-ből származik, ezért öröklődnek a híváslista importálásához szükséges metódusok. ACustomerEntry osztály fogja tartalmazni a híváslista importálása után az ügyfélre vonatkozó adatokat, az importálandó csv fájlban a név és a telefonszám szerepel vesszővel elválasztva. A beolvasott elemek egy ObservableCollection-be kerülnek, így a GUI-n is megjelennek a DataGrid-ben az ügyfelekre vonatkozó bejegyzések, ami a hívások aktuális állapotát mutatja.

## AgentDialer

Az AgentDialerosztály végzi az ügyféllista alapján a hívások kezelését és ütemezését.Ősosztálya a Scheduler<CustomerEntry>, így könnyen felhasználhatók az ütemezéshez szükséges funkciók. A StartWorks(ObservableCollectionEx<CustomerEntry> observableCollectionEx)metódus egy új szálon hívja meg aInternalStart(ObservableCollectionEx<CustomerEntry> observableCollectionEx)metódust. Itt az OpsClient-ből az IClient burkolatán keresztül lekéri az Ozeki Phone System-ből azt az API extension-t, amin az azonosítóját a settings-ben megadtuk. Sikeres lekérés esetén végigiterál az ügyféllistán, átugorja a befejezett és a RetryStates listából kihagyott állapotú hívásokat, mert a hívási folyamat az iterálás végeztével újraindítható.A soron következő ügyfelet a megadott telefonszámán elkezdi hívni az API extensiont, miután létrehozta a hívást.

ICallcall = apiExtension.CreateCall(customer.PhoneNumber);

…

call.Start();

A hívás állapotának változásáról értesülnünk kell, így még elindítás előtt fel kell iratkozni a CallStateChanged és CallError eseményekre az ICall-ban. Továbbá mielőtt elindul a hívás, az ügyfélnél beállítjuk, hogy folyamatban van a felhívása, így ez az információ a GUI-n is megjelenhet a binding és az ObservableCollection használata miatt. Ebben a Task-os szálban ha elértük a párhuzamos hívás indítási limitet, akkor várakozunk egy AutoResetEvent Set-jére, amíg egy hívás le nem zárul.

AzApiCallStateChanged()metódusban ha a hívás CallState.InCallállapotba kerül, tehát az ügyfél fogadta a hívást, kiválasztjuk az ügynökök közül azt, aki éppen szabad állapotú. Beállítástól függően figyelembe vesszük azt is, hogy az ügynököknek eddig hány hívásuk volt, és a legkevesebb hívással rendelkező ügynököt választjuk ki. Ezután az aktív hívást az API extension és az ügyfél között áttranszferáljük a kiválasztott ügynöknek, majd hozzárendeljük a kiválasztott AgentEntry-t a CustomerEntry-hez, amelyet hívás azonosító alapján tudunk elérni az eltárolt ObservableCollection-ből.

var target = handlingAgent.AgentInfo.PhoneNumber;

call.BlindTransfer(target);  
var handledCustomer = Customers.FirstOrDefault(customer => customer.CallId == call.CallId);

Abban az esetben, ha a hívás állapota CallState.Completed, és nem található a CustomerEntry –hez ügynök, a hívás véget ért, még mielőtt az ügynökhöz transzferálhatta volna az AgentDialer. Még az AgentDialer konstruktorában feliratkoztunk az OpsClient SessionCompletedeseményére. Így a ClientOnSessionCompleted metódusban kapunk értesítést arról, hogy az ügyfél és az ügynök összekapcsolása milyen állapotváltozás történt ebben a session-ben. Ez azért szükséges, mert transzferálás után az API extension hívása megszűnik, és ezután már csak a session-ök keresztül kapunk értesítést arról, hogy a két fél között a hívás végetért. A két fél között lezárult hívást úgy különböztetjük meg a többitől, hogy megnézzük

* benne van-e valamelyik ügynök mellékei között a lezárt hívás hívó azonosítója
* nincs-e több olyan aktív session, amiben a fent megtalált ügynök melléke szerepelne, tehát nincs több hívása az ügynöknek

Amennyiben ezek a feltételek teljesültek, a kérdéses hívást megtaláltuk, ezt követően a CallEnded(AgentEntry agent, CustomerEntry customer, ProgressState state) hívódik meg, ami

* Frissíti az ügynök állapotát (visszaállítja szabadra), ha a lezárt hívás már ügynökhöz került.
* Frissíti az ügyfél állapotát az átadott paraméter alapján, ami a GUI-n is megjelenik.
* Az AutoResetEvent-en meghívja a Set()-et, így vizsgálat után továbbhaladhat a korábbi iteráció az ügyfeleken