

# **Relatório técnico de implementação da CTIToolbar**

## **Contact Center Radiomóvel**

Autor: Fernando Miguel Carvalho



## Índice

<b>Relatório técnico de implementação da CTIToolbar .....</b>	<b>1</b>
<b>Contact Center Radiomóvel .....</b>	<b>1</b>
○ <b>Introdução .....</b>	<b>5</b>
○ <b>Definições .....</b>	<b>5</b>
○ <b>Enquadramento .....</b>	<b>6</b>
○ <b>Nomenclatura .....</b>	<b>9</b>
○ <b>CTI Server .....</b>	<b>10</b>
○ <b>CTIToolbar .....</b>	<b>11</b>
○ <b>AgentsManager .....</b>	<b>14</b>
○ <b>Comunicação CTIToolbar &lt;&gt; AgentsManager .....</b>	<b>16</b>
▪ <b>Diagramas de Classes .....</b>	<b>16</b>
▪ <b>Configuração <i>remoting</i> .....</b>	<b>19</b>
○ <b>SocketListener .....</b>	<b>22</b>
○ <b>AILClientLibrary .....</b>	<b>25</b>
○ <b>CTIToolbar e AILControl .....</b>	<b>29</b>
○ <b>Modo de Funcionamento .....</b>	<b>32</b>
○ <b>Funcionalidades .....</b>	<b>34</b>
▪ <b>Login e Logout .....</b>	<b>35</b>
▪ <b>Ready e NotReady .....</b>	<b>36</b>
▪ <b>Estado do Agente .....</b>	<b>37</b>
▪ <b>Customer Center .....</b>	<b>37</b>
▪ <b>Linha de Atendimento e Opção no IVR .....</b>	<b>37</b>
▪ <b>Tempo da chamada em espera .....</b>	<b>38</b>
▪ <b>Duração da chamada .....</b>	<b>38</b>
▪ <b>Atender e Desligar .....</b>	<b>38</b>
▪ <b>Ligar .....</b>	<b>38</b>
▪ <b>Chamada em Espera (<i>Hold</i>) e Recuperação .....</b>	<b>39</b>
▪ <b>Fim de Sessão .....</b>	<b>39</b>
▪ <b>Consulta .....</b>	<b>39</b>
▪ <b>Transferência .....</b>	<b>40</b>
▪ <b>Consulta para Conferência .....</b>	<b>40</b>
▪ <b>Estabelecimento de Conferência .....</b>	<b>40</b>
▪ <b>Abandono de Conferência .....</b>	<b>40</b>
<b>- Anexo – Manual de Utilizador do <i>AgentsManager</i> .....</b>	<b>41</b>

## Índice de Figuras

Figura 1 – Arquitectura base de um <i>contact center</i> .....	6
Figura 2 – Fluxo de uma operação telefónica a partir de um agente.....	7
Figura 3 – Modelo de interacção da CTIToolbar e AgentsManager no <i>contact center</i> .....	7
Figura 4 – Máquina de estados do agente.....	8
Figura 5 – Máquina de estados de uma sessão.....	8
Figura 6 – Arquitectura e modelo de integração do AIL Agent Server .....	10
Figura 7 – Arquitectura da CTIToolbar .....	12
Figura 8 – Arquitectura do AgentsManager.....	14
Figura 9 – Especificação de <i>IAgentsRemoteServer</i> e <i>IAgentRemoteClient</i> .....	17
Figura 10 – Eventos da interface <i>IAgentsRemoteServer</i> .....	17
Figura 11 – <i>AgentsRemoteClientRepeater</i> .....	18
Figura 12 – Modelo do canal de ligação via <i>.net remoting</i> [1].....	20
Figura 13 – Diagrama de classes de <i>SocketListener</i> .....	23
Figura 14 – <i>SocketServer user interface</i> .....	24
Figura 15 – Diagrama de classes para representação um evento AIL.....	26
Figura 16 – Eventos de <i>AILClient</i> .....	27
Figura 17 – CTIToolbar em modo <i>Dock</i> .....	32
Figura 18 – CTIToolbar em modo <i>Float</i> .....	33
Figura 19 – Ícone de execução da CTIToolbar na <i>notification area</i> .....	33
Figura 20 – Menu da CTIToolbar .....	33
Figura 21 – Exemplo de uma ToolTip: campo de texto para a extensão telefónica .....	34
Figura 22 – <i>Layout</i> da CTIToolbar e respectivas funcionalidades.....	35
Figura 23 – Form para introdução do <i>Username, Password e Queue</i> .....	36
Figura 24 – Botão de Login e Logout.....	36
Figura 25 – Botão de Ready e NotReady .....	36
Figura 26 – NotReady Reasons.....	37
Figura 27 – Estados do Agente .....	37
Figura 28 – Customer Center .....	37
Figura 29 – Linha de Atendimento e Opção no IVR .....	38
Figura 30 – Tempo de espera e duração da chamada .....	38
Figura 31 – Atender e Desligar .....	38
Figura 32 – Estabelecimento de uma chamada.....	39
Figura 33 – Chamada em Espera ( <i>Hold</i> ) e Recuperação .....	39
Figura 34 – Fim da Sessão.....	39
Figura 35 – Consulta e apresentação dos agentes disponíveis .....	39
Figura 36 – Transferência.....	40
Figura 37 – Consulta para Conferência.....	40
Figura 38 – Estabelecimento de Conferência.....	40
Figura 39 – Abandono de Conferência .....	40
Figura 40 – Layout do <i>AgentsManager</i> .....	41

## 1. Âmbito

### ○ Introdução

No contexto de uma aplicação distribuída o projecto de implementação de um **Sistema de controlo de Agentes num Contact Center** envolve o desenvolvimento de um componente cliente (**CTIToolbar**) e um componente servidor (**AgentsManager**), sobre o *middleware* Microsoft .net.

No capítulo 1 **Âmbito** são dados, resumidamente, os principais conceitos e entidades envolvidas na infraestrutura tecnológica de um *contact center*. No modelo do *contact center* é explicado de que forma é que se enquadram a **CTIToolbar** e o **AgentsManager**.

No capítulo 2 **Arquitectura** são traçadas as linhas de desenvolvimento para estes dois componentes e a respectiva arquitectura, descrevendo as razões que fundamentam as opções tomadas.

No capítulo 3 **Requisitos Técnicos** são descritos os detalhes da implementação ao nível da concepção e interligação entre os vários módulos que compõem o sistema.

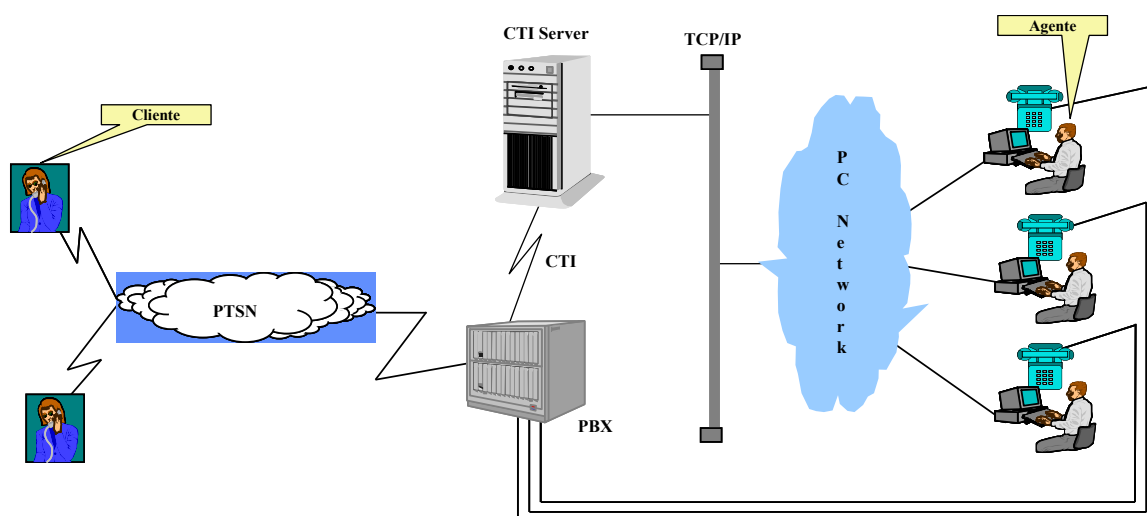
### ○ Definições

- Agente – Operador que faz o atendimento de chamadas telefónicas no *contact center*.
- ACW – *After call work*. Período de tempo que poderá decorrer entre duas chamadas atendidas pelo mesmo agente.
- API – *Application programing interface*.
- CTI – *Computer telephony integration*.
- CRM – *Contact Relationship Management*.
- PBX – Central telefónica
- PG – *Processing Group*. Grupo de agentes que recebe chamadas telefónicas de uma determinada linha de atendimento.
- Proxy – *Stub* do lado cliente com a mesma API do servidor.
- PSTN – *Public Switch Telephony Network*
- Scion – *Stub* do lado servidor que recebe os pedidos feitos pelo proxy e processa-os em invocações ao servidor.
- Wsdl – *Web Service Definition Language*.

## ○ Enquadramento

O **contact center** é a parte de uma organização que faz a gestão da **relação e contactos** com o cliente. No âmbito deste projecto apenas é considerado o contacto telefónico, embora esteja também subjacente ao contexto de um **contact center** outro tipo de contactos com suporte de outras redes públicas como a Internet. O **Agente** é a entidade do **contact center** que representa a organização na comunicação telefónica com o cliente.

Fazem parte deste sistema distribuído as entidades referidas na **Figura 1**.

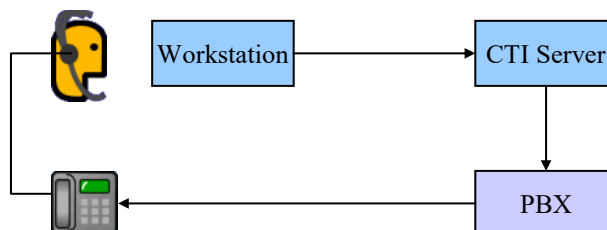


**Figura 1 – Arquitectura base de um *contact center***

Destacam-se da arquitectura base de um **contact center** as seguintes entidades:

- PBX – Central telefónica ligada à rede pública (PSTN) que faz a recepção, gestão e distribuição de todas as chamadas telefónicas que entram, saem e ocorrem no **contact center**.
- Agente – Recebe ou estabelece a chamada telefónica com o Cliente. Tipicamente um agente tem como infra-estrutura de suporte ao seu trabalho, uma **workstation** e uma **extensão telefónica** ligada ao PBX.
- CTI Server – Tem uma ligação com a rede local das **workstations** dos agentes e uma ligação CTI (**Computer Telephony Integration**) com o PBX, que poderá ser suportada, ou não, pela mesma rede local.

Desta forma, todas as operações telefónicas efectuadas pelo agente são comandadas pelo servidor CTI, não havendo interacção directa do agente com a sua extensão telefónica. Quer isto dizer que uma determinada operação sobre uma chamada telefónica é desencadeada pelo agente a partir da sua workstation conforme apresenta a **Figura 2**.



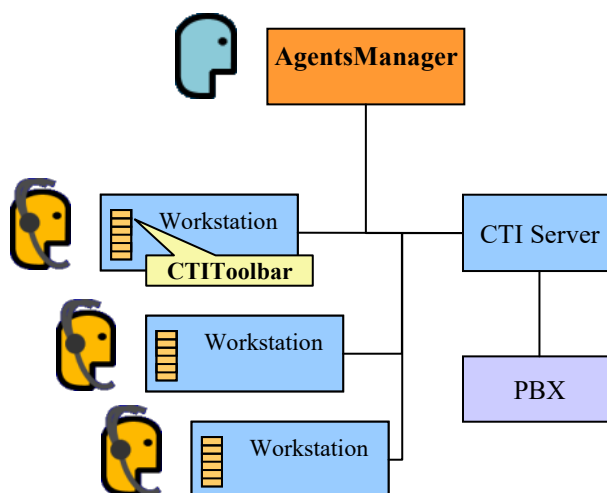
**Figura 2 – Fluxo de uma operação telefónica a partir de um agente**

Do mesmo modo, os eventos sobre as chamadas telefónicas chegam ao agente através de um fluxo inverso ao da Figura 2.

Neste modelo a componente *CTIToolbar* e o servidor *AgentsManager* têm os seguintes papéis:

- **CTIToolbar** – Aplicação cliente que corre na *workstation* do agente e faz a integração com o *CTI Server*.
- **AgentsManager** – Aplicação de supervisão que recebe ligações de todas as *CTIToolbar* sendo notificado do estado dos respectivos agentes e disponibilizando um conjunto de operações sobre estes.

Tipicamente cada agente utilizará como ferramenta de trabalho a *CTIToolbar* enquanto um supervisor fará o controlo da actividade dos agentes através do *AgentsManager*, de acordo com o modelo da Figura 3.



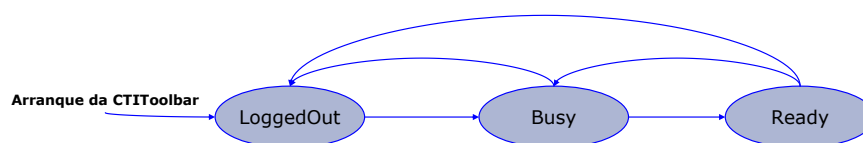
**Figura 3 – Modelo de interacção da CTIToolbar e AgentsManager no contact center**

Toda a actividade laboral desenvolvida pelo agente no *contact center* é centralizada pela *CTIToolbar*. Durante a sua actividade o agente irá transitar entre vários estados destacando-se os seguintes:

- *Logoff* ou *Logged Out* – Estado inicial do agente quando arranca a *CTIToolbar*.

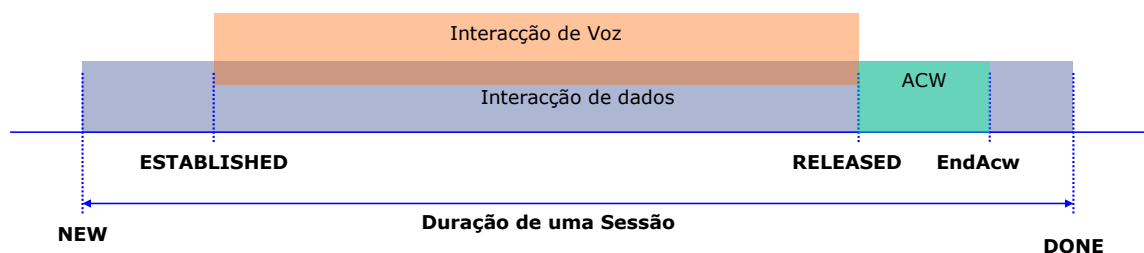
- *Busy* – Estado do agente depois de efectuar o *login*. Se a operação de login for bem sucedida a *CTIToolbar* estabelecerá as ligações ao *CTI Server*, *AgentsManager* e fará a ligação a um grupo de atendimento (PG) na central telefónica. Neste estado o agente está ligado ao servidor, mas não disponível para receber chamadas do exterior.
- *Ready* – O agente está ligado e disponível para receber chamadas do exterior.

A transição de estado do agente obedece à máquina de estados da **Figura 4**:



**Figura 4 – Máquina de estados do agente**

Sempre que um agente recebe uma chamada telefónica do exterior é estabelecida uma **sessão** (estado *working*). Uma sessão integra uma ou duas interações: voz e dados, conforme apresenta a **Figura 5**.



**Figura 5 – Máquina de estados de uma sessão**

Os eventos de NEW, ESTABLISHED, RELEASED e DONE, marcam respectivamente o início da sessão, da chamada telefónica, o final da chamada telefónica e da sessão. Entre o fim da chamada e da sessão decorre o período de ACW (*After Call Work*). O ACW é o período máximo que o Agente pode garantir sem receber chamadas durante o seu estado de *Ready*. Terminado este período o sistema poderá entregar uma chamada ao agente mesmo que este ainda esteja *working*, ou seja, não tenha finalizado a sessão.

Todas as transições de estado e de sessões do agente são monitorizadas através do servidor *AgentsManager*. Este servidor poderá ainda actuar sobre os agentes mudando-os de estado ou ainda de grupo de atendimento no *contact center* (mais detalhes funcionais podem ser consultados em - **Anexo – Manual de Utilizador do *AgentsManager***). Assim, o servidor *AgentsManager* será especificamente utilizado por um supervisor do *contact center*.

Os agentes, além de terem a *CTIToolbar* para controlo das chamadas telefónicas, utilizarão ainda como ferramenta de trabalho uma aplicação de CRM (*Contact Relationship Management*), com a designação de **Kenan Customer Center**. Sempre que é recebida uma chamada telefónica a *CTIToolbar* enviará informação para o *Customer Center*, para que seja automaticamente apresentada a ficha de dados associada ao cliente.



## 2. Arquitectura

O sistema distribuído apresentado no capítulo -o **Enquadramento** integra 4 aplicações distintas *AgentsManager*, *CTIToolbar*, *Kenan Customer Center* e *CTI Server*. Fazem parte do âmbito deste projecto o desenvolvimento do *AgentsManager* e da *CTIToolbar*.



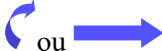
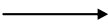


O desenho da *CTIToolbar* e *AgentsManager* além de seguir os requisitos de uma aplicação distribuída é fortemente delineado pelas características do ambiente em que se integram, o *contact center*, que envolve o *CTI Server* e o *Kenan Customer Center*.

Neste capítulo serão descritos, pela seguinte ordem:

- *CTI Server* – Modelo de integração com a aplicação dos agentes (*CTIToolbar*);
- *CTIToolbar* – Arquitectura e integração com o *CTI Server*;
- *AgentsManager* – Arquitectura e integração com os clientes, *CTIToolbar*.

### ○ Nomenclatura

Nas especificações apresentadas neste capítulo são usados modelos que representam a arquitectura de alguns dos componentes. Nestes modelos são usadas figuras com o seguinte significado:

-  - Processo
-  - Instância
-  - Evento
-  - Invocação de um método remoto ou transmissão de dados via *socket*.
-  - Exemplo de uma ligação a um porto 11095.
-  - Referência para um objecto remoto.

## ○ CTI Server

O *contact center*, como parte integrante de uma organização, além de assentar na infra-estrutura tecnológica apresentada no capítulo -o **Enquadramento**, tem de coexistir simultaneamente com outros sistemas. Isto faz com que a simplicidade do processo apresentado na **Figura 2** se torne mais complexo devido:

- Ao **número** de aplicações que coexistem na workstation do agente;
- À **heterogenidade** existente entre as aplicações.

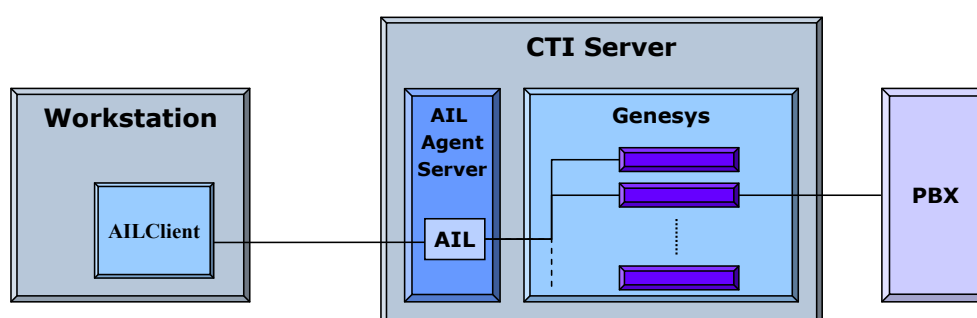
Por isso é usual que os servidores de CTI existentes no mercado não sejam peças monolíticas, mas também um sistema distribuído. Ou seja, a sua arquitectura baseia-se num conjunto de componentes com funcionalidades distintas que poderão ser distribuídos por várias máquinas. Logo, a imagem do servidor CTI apresentada na **Figura 1** trata-se apenas de uma das possíveis abordagens.

Simultaneamente, outro dos requisitos que o servidor CTI tem que satisfazer é a disponibilização de uma API flexível, que permita um bom grau de integração na *workstation* do agente.

O **Genesys** é o servidor CTI usado neste projecto e o **AIL** – *Agent Interaction Layer* - é o componente que como o nome indica disponibiliza uma API para utilização das funcionalidades telefónicas do agente. A **AIL** é formada por um conjunto de bibliotecas em Java.

Para servir o conjunto de agentes que existem no *contact center* foi necessário acrescentar mais uma *layer* (camada) sobre a AIL que servisse eficientemente os pedidos dos agentes. A concepção deste *layer* teve que ter em conta os aspectos de heterogenidade da workstation do agente, sem comprometer opções futuras nas tecnologias a usar do lado do agente.

Assim, a opção caiu sobre uma aplicação J2EE que disponibiliza a mesma interface da AIL e que é designada neste documento por **AIL Agent Server**.



**Figura 6 – Arquitectura e modelo de integração do AIL Agent Server**

O *AIL Agent Server* disponibiliza num *web service* designado **AILAgent** um conjunto de métodos através dos quais os agentes podem invocar operações telefónicas.

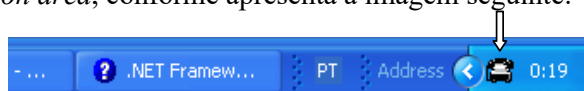
Esta solução dá um bom grau de liberdade na concepção das aplicações do agente. Contudo um canal de comunicação sobre HTTP tem como obstáculo a notificação de eventos do servidor para o cliente.

Para contornar este problema, o *AIL Agent Server* permite o estabelecimento de um socket TCP por agente, através do qual são enviados os eventos do servidor.

## ○ CTIToolbar

A *CTIToolbar*, que é o Cliente AIL representado na **Figura 6**, tem que satisfazer os seguintes requisitos:

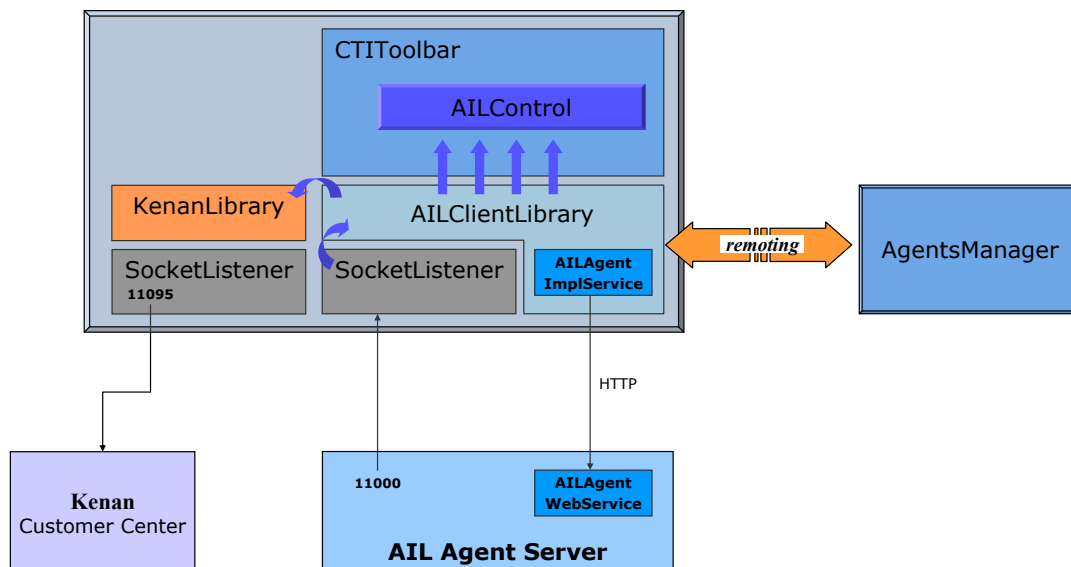
- Comportamento de uma aplicação **Desktop Toolbar**. Quer isto dizer, que poderá ser alojada (*dock*) num dos lados do desktop não podendo ser aquele espaço usado por nenhuma outra aplicação (semelhante à *Windows Taskbar*). Enquanto o Agente não tem um contacto a decorrer a aplicação poderá ser minimizada ficando em execução em *background*, sendo apenas visível o seu ícone de execução na *notification área*, conforme apresenta a imagem seguinte:



(Mais detalhes funcionais desta aplicação poderão ser consultados no capítulo 6 **Anexo – Manual de Utilizador da CTIToolbar**)

- A *user interface* da *CTIToolbar* deve ser implementada num *control .net* para que possa posteriormente ser reutilizado noutro tipo de aplicação, que não a *CTIToolbar*.
- Integração com o *AIL Agent Server* através do *web service*, *AILAgent* e do socket para a receção de eventos.
- Integração com o servidor *AgentsManager* notificando-o sobre o estado deste agente e processando os pedidos enviados pelo servidor que operam sobre a *CTIToolbar*.
- Integração com o *Kenan Customer Center* por intermédio de um *socket*, cuja ligação é pedida por este à *CTIToolbar* e através da qual esta lhe enviará os dados do cliente associado a uma chamada telefónica.

Estes requisitos conduziram à concepção da *CTIToolbar* com a arquitectura apresentada na **Figura 7**.



**Figura 7 – Arquitectura da CTIToolbar**

Destacam-se do modelo apresentado na **Figura 7**:

- **CTIToolbar** - Aplicação *Desktop Toolbar* que integra o controlo **AILControl**.
- **AILControl** – **Control** .net que disponibiliza as operações telefónicas ao agente.
- **AILClientLibrary** – Biblioteca .net que disponibiliza numa API o conjunto de operações telefónicas que são integradas no **AILControl**. Nesta *interface* é ainda mantido um conjunto de eventos que traduzem o estado do agente na central telefónica. Por intermédio de *handlers* registados nesses eventos, a **AILControl** mantém o seu estado actualizado.  
O acesso e comunicação do servidor **AgentsManager** com os agentes é feito através desta camada.
- **SocketListener** – Faz o controlo de uma ligação via socket mantendo um conjunto de eventos que reflectem os dados recebidos e disponibilizando um conjunto de serviços para transmissão de dados. Serão usadas duas instâncias desta classe: uma para a recepção de eventos do **AIL Agent Server** e outra para ligação ao **Kenan Customer Center**.
- **AILAgentImplService** – Classe gerada a partir do *wsdl* do *web service* **AILAgent**. As instâncias desta classe apresentam uma API semelhante ao *web service* **AILAgent**, processando os pedidos recebidos em invocações ao *web service*.
- **KenanLibrary** – Recebe os eventos de início de nova sessão da **AILClientLibrary** e transmite esses dados ao **Kenan Customer Center** via *socket*.

No modelo da **Figura 7** as setas a azul representam o fluxo de eventos. Neste caso a “fonte” provem dos dados que são recebidos do *socket* que está ligado ao **CTIServer** e que notificam a **CTIToolbar** sobre o estado do agente e das sessões. Os dados recebidos são desserializados em objectos que são transmitidos nos eventos lançados pela **AILClientLibrary**.

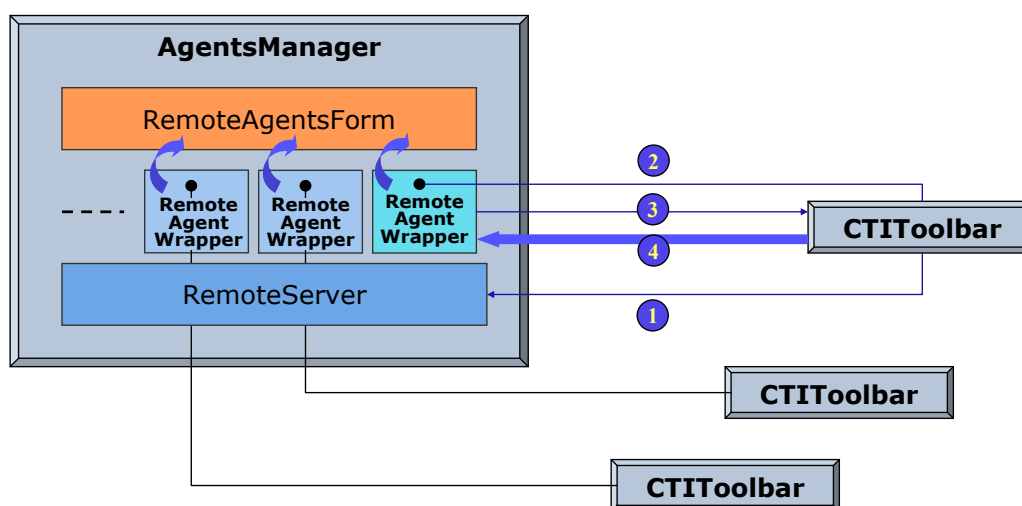
A *AILControl* tem *handlers* sobre estes eventos de modo a manter actualizados os estados dos botões enquanto que a *KenanLibrary* ao ser notificada do início de uma nova sessão vai transmitir essa informação ao *Kenan Customer Center* para que seja apresentado na *workstation* do agente a informação do cliente.

Dos componentes que integram a arquitectura da *CTIToolbar* será dado ênfase à descrição técnica *AILClientLibrary* e do *SocketListener* nos capítulos -o e -o, respectivamente.

## ○ AgentsManager

O servidor *AgentsManager* consiste numa .net *Windows Application* que tem registado na infra-estrutura .net *remoting* um *well known service* ao qual se ligam os vários clientes *CTIToolbar*.

Os serviços acessíveis remotamente são especificados pela interface **IAgentsRemoteServer** e implementados pelo **RemoteServer** no servidor *AgentsManager*. Neste caso, a *CTIToolbar* que se liga ao *AgentsManager* registar-se-á neste servidor através da invocação de um método remoto, passando-lhe a sua referência como parâmetro (**Figura 8**, passo 1). Por sua vez, o servidor *AgentsManager* ficará com uma referência remota para cada uma das *CTIToolbar*'s que estiverem a si ligadas (**Figura 8**, passo 2), através da qual poderá efectuar operações sobre estas (**Figura 8**, passo 3). Por cada uma destas ligações é mantido no servidor uma instância **RemoteAgentWrapper**.



**Figura 8 – Arquitectura do AgentsManager**

Além disso o servidor *AgentsManager* registará ainda um conjunto de *handlers* sobre determinados eventos da *CTIToolbar* que notificam sobre o estado em que o agente se encontra (**Figura 8**, passo 4). Desta forma, o *AgentsManager* manterá actualizado a vista sobre os vários agentes.

O estabelecimento do fluxo descrito na **Figura 8**, obriga a um conjunto de configurações adicionais do *AgentsManager* e *CTIToolbar* que serão detalhadas no capítulo -o **Comunicação CTIToolbar <> AgentsManager**.

Conforme apresenta a **Figura 8** a arquitectura do *AgentsManager* está organizada nas seguintes camadas:

- **Remote Server** – Implementação da interface **IAgentsRemoteServer** que tem o serviço registado na infra-estrutura .net *remoting* como “*AgentsManagerRemoteServerURI.rem*”.
- **RemoteAgentsForm** – .net *Windows Application* mantém uma vista actualizada do estado dos agentes no *contact center*, transmitida pelas respectivas instâncias de *CTIToolbar* de cada agente. Permite ainda efectuar operações sobre estes agentes.

- **RemoteAgentWrapper** – Guarda uma referência remota para cada instância de **CTIToolbar** e para o “item” correspondente na **RemoteAgentsForm**, fazendo o mapeamento entre os eventos recebidos e o item a actualizar na vista de agentes. No sentido inverso, faz o encaminhamento das operações em invocações remotas à **CTIToolbar**.

### 3. Requisitos Técnicos

Neste capítulo serão abordados os aspectos técnicos da implementação do **Sistema de controlo de agentes num *contact center***, dando maior profundidade a assuntos mais directamente ligados à cadeia de ARGE

Não serão mais detalhadas as funcionalidades das aplicações envolvidas, além daquilo que já foi descrito nos capítulos anteriores. Contudo, esclarecimentos sobre os aspectos funcionais das aplicações podem ser obtidos nos capítulos, 6 Anexo – Manual de Utilizador da CTIToolbar e - **Anexo – Manual de Utilizador do *AgentsManager***.

Serão descritos pela seguinte ordem:

- -o **Comunicação CTIToolbar <> AgentsManager**
- -o **SocketListener**
- -o **AILClientLibrary**
- -o **CTIToolbar e AILControl**

#### ○ **Comunicação CTIToolbar <> AgentsManager**

Tal como foi explicado ao longo dos capítulos anteriores o servidor ***AgentsManager*** recebe a informação de estado de todos os clientes ***CTIToolbar*** e permite ainda actuar sobre estes últimos através da invocação remota de algumas operações.

Uma vez estabelecida a ligação *.net remoting*, a invocação de operações e recepção de eventos da ***CTIToolbar*** são feitos através da camada ***AILClientLibrary*** apresentada no capítulo, -o **CTIToolbar**. Ou seja, numa perspectiva de processo remoto, o ***AgentsManager*** tem um comportamento semelhante ao do ***AILControl*** dentro do processo da ***CTIToolbar***.

Com base nesta ideia, a utilização do *.net remoting* na ligação entre o ***AgentsManager*** e a ***CTIToolbar***, tem como objectivo minimizar o aumento de funcionalidades do lado da ***CTIToolbar*** para o estabelecimento do fluxo de interações entre os dois componentes, conforme foi detalhado no capítulo -o **AgentsManager**.

No próximo sub capítulo, -o □ **Diagramas de Classes**, são explicados, quais os componentes adicionais na arquitectura da ***CTIToolbar*** e ***AgentsManager*** que permitem o estabelecimento da comunicação via *.net remoting* e de que forma as opções tomadas minimizam a interdependência das duas aplicações.

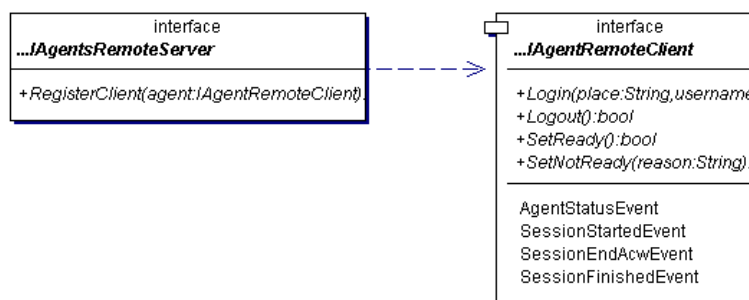
No sub capítulo, -o □ **Configuração *remoting***, são descritas as configurações de ambiente necessárias ao estabelecimento dessa ligação.

#### ▪ **Diagramas de Classes**



Os eventos registados e os métodos invocados pelo *AgentsManager* são os mesmos que são usados pelo *AILControl*. Para tal é necessário publicar esses eventos e métodos da *AILClientLibrary* como sendo acessíveis remotamente. Esses métodos e eventos foram especificados numa interface **IAgentRemoteClient**.

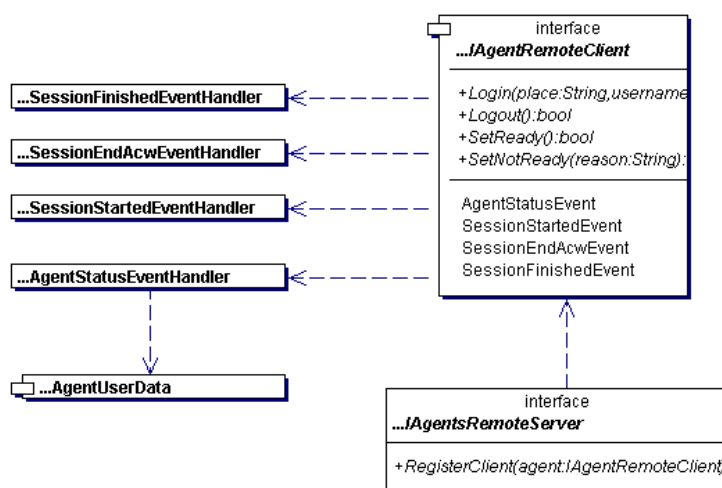
Por sua vez a API do *AgentsManager* acessível via *remoting* é especificada pela interface **IAgentsRemoteServer**.



**Figura 9 – Especificação de IAgentsRemoteServer e IAgentRemoteClient**

Uma vez que a *metadata* de ambas as interfaces tem que estar acessível aos dois componentes *AgentsManager* e *CTIToolbar*, foi definida uma biblioteca num *assembly* separado (**AgentRemoteLibrary.dll**) de forma a evitar a interdependência entre as implementações destes componentes.

O servidor *AgentsManager* interage com os clientes *CTIToolbar* pela invocação de métodos remotos e pela recepção de eventos definidos na interface **IAgentRemoteClient**. Estes eventos são implementados através de *delegates* .net, conforme apresentado na **Figura 10**.



**Figura 10 – Eventos da interface IAgentsRemoteServer.**

Um *delegate* é um tipo.net que define um ponteiro para função. Um evento do tipo *delegate* pode ser visto como uma lista de ponteiros para funções, que estão registadas como *handlers* desse evento.

Assim, uma função ao ser registada como *handler* de um evento, vai passar como parâmetro ao evento um ponteiro para o seu endereço na tabela de funções ou métodos virtuais. Do lado do evento, a criação de uma instância (do tipo *delegate*) que vai encapsular o ponteiro para uma função, requer o acesso à *metadata* dessa função.

Uma vez que os *handlers* para os eventos da *CTIToolbar* serão implementados ao nível do *AgentsManager*, esta situação obriga a que a *CTIToolbar* tenha acesso à *metadata* do *AgentsManager* e assim ao seu *assembly*, o que cria uma interdependência não desejada entre as duas aplicações.

Mantendo o objectivo inicial de que as interfaces remotas são definidas num *assembly* separado (*AgentRemoteLibrary.dll*) e usando uma solução baseada num *EventRepeater* conforme apresentada em [1], foi adicionada uma classe **AgentRemoteClientRepeater** que fará o mapeamento entre os eventos de **IAgentRemoteClient** e os *handlers* implementados na aplicação *AgentsManager*.

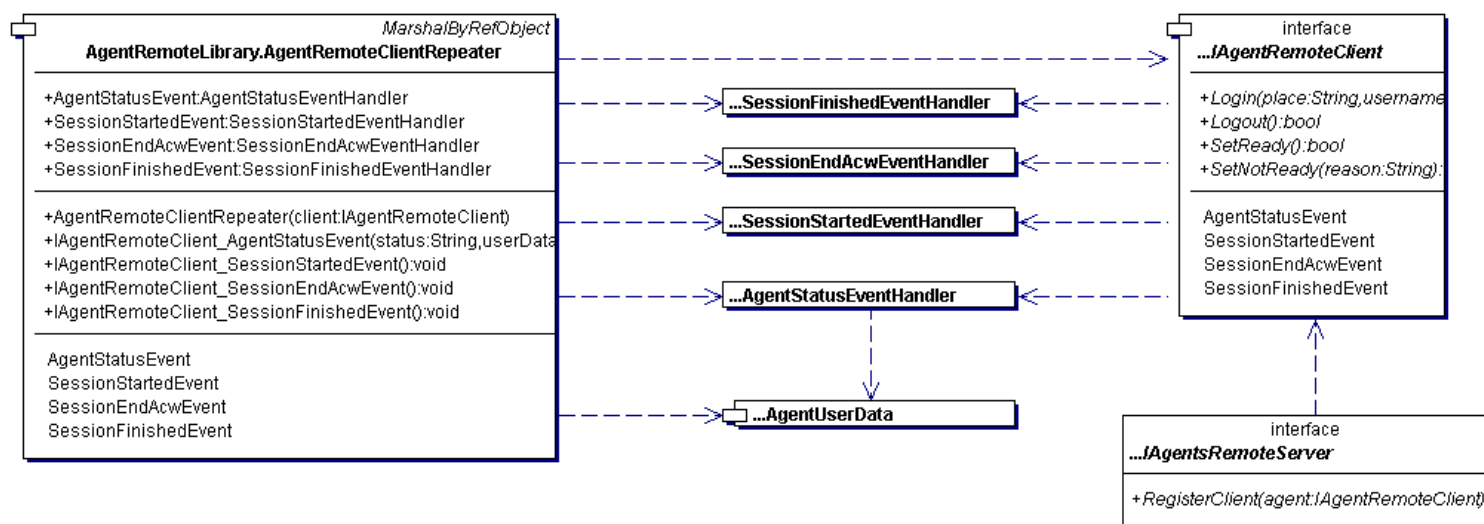


Figura 11 –AgentsRemoteClientRepeater

Conforme apresentado em -o **AgentsManager**, no servidor *AgentsManager* existirá uma instância de **RemoteAgentWrapper** por cada *CTIToolbar*, que esteja a si ligado. Esta instância de **RemoteAgentWrapper** terá os *handlers* para os eventos de **IAgentRemoteClient**, que de acordo com o modelo agora apresentado serão registados sobre uma instância de **AgentRemoteClientRepeater**.

Resumindo os 4 passos da interacção entre o *AgentsManager* e *CTIToolbar* dão-se da seguinte forma:

1. Uma instância da *CTIToolbar* é inicializada e passa a sua referência ao *AgentsManager* através da invocação do método **RegisterClient** de **IAgentsRemoteServer**.
2. O *AgentsManager* cria uma instância:
  - **AgentRemoteClientRepeater** para a referência da *CTIToolbar* recebida.
  - **RemoteAgentWrapper** que registará um conjunto de *handlers* sobre os eventos de **AgentRemoteClientRepeater**.

3. O *AgentsManager* pode fazer invocações à *CTIToolbar* através dos métodos da *IAgentRemoteClient*.
4. Os eventos da *CTIToolbar* são enviados ao **AgentRemoteClientRepeater** que por sua vez serão encaminhados para o **RemoteAgentWrapper** que actualizará o estado do agente no *AgentsManager*.

## ▪ Configuração *remoting*

A configuração do *AgentsManager* sobre o .net *remoting* é feita via programação do **RemoteServer**, classe que implementa a interface **IAgentsRemoteServer**. Por sua vez, a configuração da *CTIToolbar* é feita através do ficheiro de configuração, do tipo *app.config*. Deste modo foi adquirido conhecimento sobre as duas formas de configuração.

Uma outra razão para se usar dois modos de configuração distintos tem a ver com a diferença conceptual entre o *AgentsManager* e a *CTIToolbar*. Enquanto que o objectivo do *AgentsManager* é servir os vários clientes, a *CTIToolbar* poderá desempenhar o seu papel sem depender da existência do *AgentsManager*.

Na *CTIToolbar*, a classe que implementa a interface **IAgentRemoteClient** é a **AILClient** e a sua configuração reside no ficheiro *AILClientLibrary.config*. Neste ficheiro a ligação *remoting* ao servidor *AgentsManager* é especificada na marca `system.runtime.remoting`. Aqui são definidos através dos atributos:

- *type* – Definição do tipo remoto na forma:  
`<namespace>.<remote type>, <assembly name>`  
neste caso:  
`AgentsManagerRemoteApp.IAgentsRemoteServer, AgentsManagerRemoteApp`
- *url* – Especificação do identificador para o tipo remoto na forma:  
`<protocolo>://<maquina>:<porto>/<NomeDoServiço>`  
neste caso:  
`url="tcp://localhost:8085/AgentsManagerRemoteServerURI.rem"`

Ao nível da especificação do *url* existe uma diferença entre a sua definição ao nível do servidor e cliente. No cliente esta identificação tem que conter obrigatoriamente o tipo de protocolo (*tcp* ou *http*) e o porto do serviço remoto. A ausência desta informação não permite o estabelecimento do canal de comunicação.

No caso do servidor (*AgentsManager*) o *url* com que este serviço é publicado deve conter apenas o nome do serviço, excluindo o protocolo e o porto.

Além disto o servidor também fará invocações de métodos do cliente. Este requisito obriga assim, a que além da especificação dos parâmetros já referidos, seja ainda especificado um canal do lado do cliente para a recepção dos pedidos do servidor, através da seguinte marca:

```
<channels>
  <channel ref="tcp" port="0">
</channels>
```

O facto de o servidor *AgentsManager* ter a possibilidade de invocar operações sobre os seus clientes, implica a criação de *proxies* para as *CTIToolbar*'s do lado do servidor. Este *proxie* é criado a partir do objecto *ObjRef* que

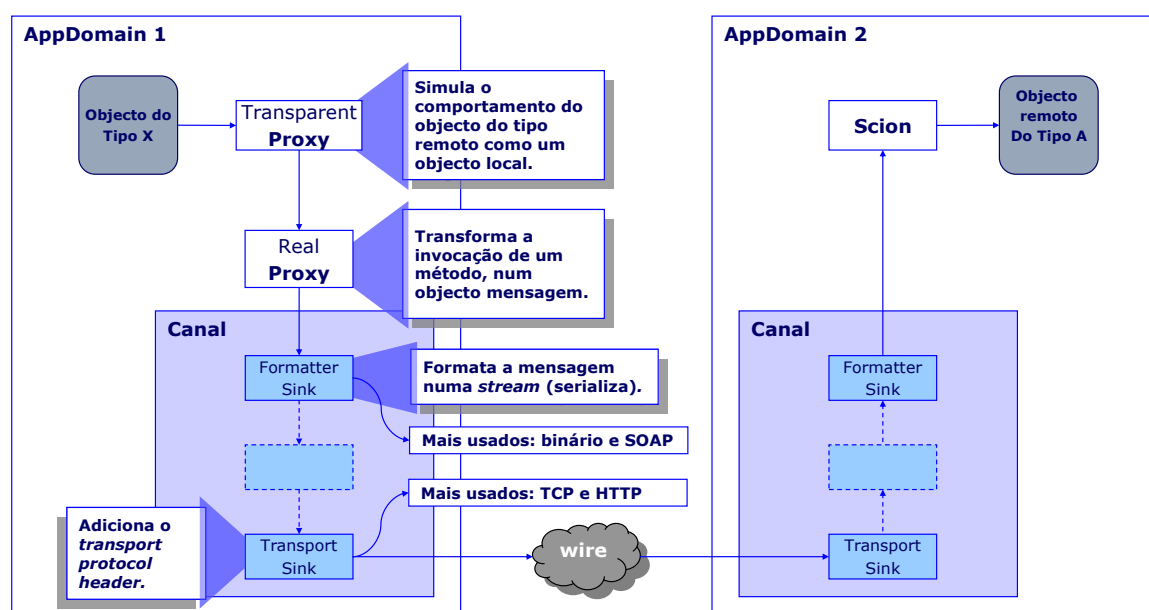
foi enviado para o servidor na invocação do método **RegisterClient** de **IAgentsRemoteServer**. Para que seja passada uma referência de uma instância de **AILClient** num objecto **ObjRef** é necessário que esta classe derive de **MarshalByRefObject**.

Contudo, embora o cliente seja do tipo **MarshalByRefObject** não é um serviço registado na plataforma .net *remoting*, à semelhança do que se passa com o servidor *AgentsManager*.

O nível de desserialização por defeito do .net *remoting* (**Low**) só permite a desserialização de tipos da infraestrutura de *remoting* e um conjunto limitado de tipos do sistema. Assim o nível de desserialização tem que ser passado para **Full**, para que possa ser desserializada a instância de **ObjRef** que contem a referência para o cliente. Caso contrário, o servidor ao tentar desserializar o objecto **ObjRef** recebido dará uma excepção do tipo **SerializationException** com a seguinte mensagem associada:

*“Because of security restrictions, the type System.Runtime.Remoting.ObjRef cannot be accessed.”*

O nível de desserialização é definido no **canal**, mais especificamente no **Formatter Sink**. O canal de comunicação entre o **proxy** e o **scion** assenta numa arquitectura flexível, formada por uma série de objectos **Channel Sink**, que se ligam entre si formando uma cadeia como apresenta a **Figura 12**. Cada um destes **Channel Sink** tem uma função específica no processamento da mensagem transmitida.



**Figura 12 – Modelo do canal de ligação via .net remoting [1]**

No caso do nível de desserialização referido anteriormente, este é controlado ao nível do **Formatter Sink**. Nesta situação em que é usada serialização em binário terá que ser usada uma instância do tipo **BinaryServerFormatterSinkProvider** alterando o atributo **TypeFilterLevel** para **Full**.

Por sua vez a **CTIToolbar** disponibiliza um conjunto de eventos que podem ser subscritos pelo *AgentsManager*. Nesta situação é o *AgentsManager* que envia à **CTIToolbar** uma instância de **ObjRef** referenciado um *delegate* para um método seu. Para que a **CTIToolbar** possa desserializar este *delegate* também terá que ter um nível de desserialização **Full**. Esta especificação é dada pela seguinte marca:

```
<serverProviders>  
  <formatter ref="binary" typeFilterLevel="Full" />  
</serverProviders>
```

No processo de desserialização do *ObjRef* para um *delegate* é necessária a *metadata* da função encapsulada nesse *delegate*. Ou seja, não basta que a **AILClient** tenha acesso à *metadata* correspondente à declaração da função, mas sim, à respectiva implementação. Esta é a razão da existência da classe **AgentRemoteClientRepeater** ao nível da biblioteca AgentRemoteLibrary.dll.

## ○ SocketListener

Enquanto que a ligação entre a *CTIToolbar* e o *AgentsManager* é suportado pela infra estrutura .net *remoting*, aproveitando o facto de ambas as aplicações serem desenvolvidas em tecnologia .net, o mesmo não acontece na comunicação com o *CTIServer* e *Kenan Customer Center* suportados na tecnologia Java.

Assim optou-se por estabelecer uma ligação via *socket* com a definição de um protocolo específico na formatação dos dados transmitidos.

As ligações via socket são mantidas por duas instâncias de **SocketListener**, classe que encapsula um conjunto de serviços sobre um porto local para controlo de ligações e transmissão de dados.

Enquanto que na ligação ao *AIL Agent Server* este é que faz a gestão das ligações que são a ele estabelecidas, no caso do *Kenan Customer Center*, é a *CTIToolbar* que tem a responsabilidade de estar à escuta de uma nova ligação que seja pedida pela outra aplicação. Assim a classe **SocketListener** foi desenhada com vista a implementar serviços que suportassem estes dois cenários.

O fluxo de operações que leva ao estabelecimento da ligação com o *AIL Agent Server* segue a seguinte ordem:

1. A instância de **AILClient** invoca o método *Init()* ao *web service AILAgent* que retorna um *ClientId*;
2. Estabelecimento da ligação por invocação do método *Connect* de **SocketListener** e envio do *ClientId* ao *AIL Agent Server*;
3. Escuta de mensagens (eventos) para este cliente, numa *thread* separada, lançada pela invocação de *StartReceivingUTF* de **SocketListener**;
4. Registo de *handlers* para os eventos de **SocketListener**: *ConnectionClosedEvent* e *DataReceivedEvent*. Sempre que é recebido um evento do *AIL Agent Server* será invocado o *handler* do evento *DataReceivedEvent*.

O fluxo de operações que leva ao estabelecimento da ligação com o *Kenan Customer Center* segue a seguinte ordem:

1. Início da escuta de pedidos de novas ligações numa *thread* separada, através da invocação de *StartAcceptingConnections* de **SocketListener**.  
Registo de um *handler* no evento *ConnectionEstablishedEvent* de **SocketListener**, que será lançado quando for estabelecida uma nova ligação com o *Kenan Customer Center*.
2. Quando houver notificação da ligação estabelecida (por intermédio do evento *ConnectionEstablishedEvent*), passa a ficar à escuta de mensagens recebidas numa *thread* separada, através da invocação de *StartReceivingLine* de **SocketListener**.
3. Registo de um *handler* para o evento *StringReceivedEvent* de **SocketListener**, que será invocado sempre que seja recebida uma nova mensagem de **SocketListener**.

Da análise destes dois fluxos verifica-se que a implementação do `SocketListener` apresenta dois modos de recepção de mensagens devido à forma como as mensagens são enviadas do *AIL Agent Server* ou do *Kenan Customer Center*.

O *AIL Agent Server* que é implementado em Java usa um `DataStream` e o método `writeUTF`, para formatar as mensagens enviadas. Este método acrescenta no início da mensagem dois *bytes* que indicam o seu tamanho. Por sua vez, o *Kenan Customer Center* marca o fim de cada mensagem com um “`\r\n`”.

A forma como a *AILClientLibrary* envia dados ao *AIL Agent Server* ou a *KenanClientLibrary* envia ao *Kenan Customer Center*, tem que obedecer a cada um dos formatos anteriores respectivos. Por isso existe duplicação dos serviços disponibilizados pelo `SocketListener` para:

- Estabelecimento e controlo de ligações;
- Recepção de mensagens;
- Envio de mensagens.

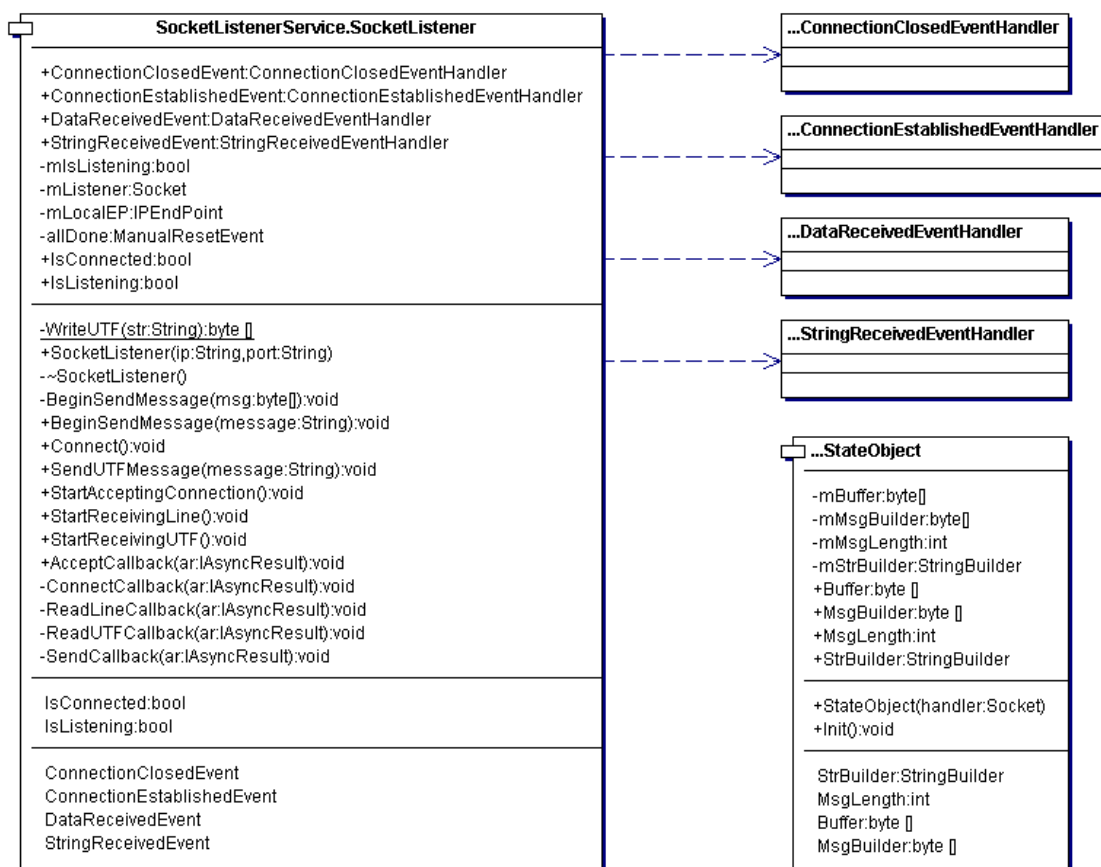


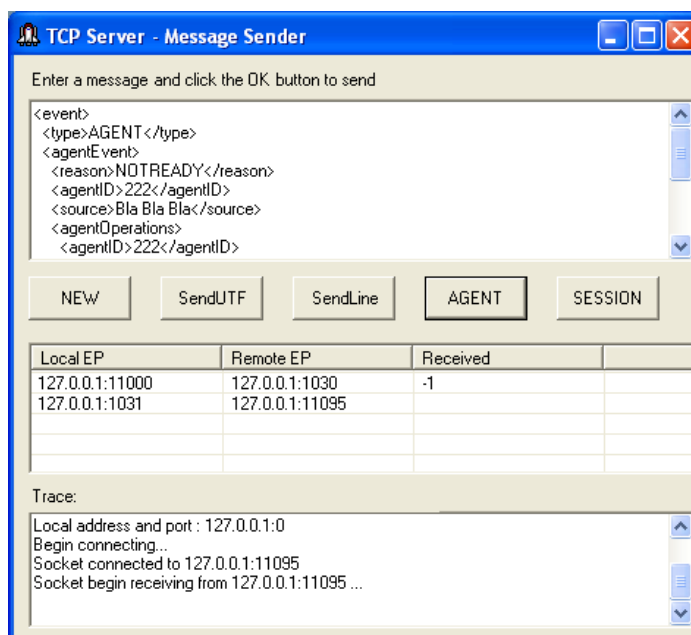
Figura 13 – Diagrama de classes de `SocketListener`

Contudo, durante a fase de desenvolvimento da *CTIToolbar*, o tratamento de eventos foi testado numa primeira etapa na ausência do servidor *AIL Agent Server* e do *Kenan Customer Center*, para tal foi desenvolvida mais uma aplicação local auxiliar, *SocketServer*, que simula estas duas aplicações através das seguintes funcionalidades:



- Recepção de pedidos e estabelecimento de novas ligações num porto 11000 (comportamento do servidor *AIL Agent Server*);
- Envio de mensagens formatadas como no método *writeUTF* da *DataStream* Java, que simulavam os eventos enviados pelo *AIL Agent Server*;
- Pedido de estabelecimento de novas ligações a um determinado porto (comportamento do *Kenan Customer Center* que faz uma ligação ao porto 11095);
- Envio de mensagens terminadas com “\r\n”, tal como no *Kenan Customer Center*;
- Tratamento das mensagens enviadas pelas ligações estabelecidas.

Todas estas funcionalidades são disponibilizadas pela *user interface* desenvolvidas na aplicação *SocketServer* conforme apresentado na **Figura 14**.



**Figura 14 – SocketServer user interface**



## ○ AILClientLibrary

A *AILClientLibrary* é a camada da *CTIToolbar* que centraliza os mecanismos de comunicação com o *CTIServer*, *AgentsManager* e *Kenan Customer Center*.

No capítulo anterior foram abordados os aspectos técnicos que suportam a interação com o *AgentsManager*. Neste capítulo será abordada a parte estrutural da *AILClientLibrary*, como e quais as invocações remotas realizadas e de que forma são tratados os eventos recebidos do *AIL Agent Server*.

No caso do *AIL Agent Server*, cujos eventos notificam sobre o estado do agente e sessões telefônicas (ver -o **Enquadramento**), as mensagens encontram-se num formato XML que obedece a cada um dos exemplos seguintes, respectivamente:

Evento de Agente:	Evento de Sessão:
<pre> &lt;event&gt;   &lt;type&gt;AGENT&lt;/type&gt;   &lt;agentEvent&gt;     &lt;reason&gt;NOTREADY&lt;/reason&gt;     &lt;agentID&gt;222&lt;/agentID&gt;     &lt;source&gt;Bla Bla Bla&lt;/source&gt;     &lt;agentOperations&gt;       &lt;agentID&gt;222&lt;/agentID&gt;       &lt;canLogout&gt;true&lt;/canLogout&gt;       &lt;canSetReady&gt;true&lt;/canSetReady&gt;       &lt;canSetNotReady&gt;false&lt;/canSetNotReady&gt;       &lt;canMakeCall&gt;true&lt;/canMakeCall&gt;       &lt;canSetACW&gt;false&lt;/canSetACW&gt;     &lt;/agentOperations&gt;   &lt;/agentEvent&gt; &lt;/event&gt; </pre>	<pre> &lt;event&gt;   &lt;type&gt;SESSION&lt;/type&gt;   &lt;sessionEvent&gt;     &lt;reason&gt;NEW&lt;/reason&gt;     &lt;sessionID&gt;123&lt;/sessionID&gt;     &lt;sessionStatus&gt;1&lt;/sessionStatus&gt;     &lt;canAnswer&gt;true&lt;/canAnswer&gt;     &lt;canCompleteConference&gt;true&lt;/canCompleteConference&gt;     &lt;canConference&gt;true&lt;/canConference&gt;     &lt;canConsultate&gt;true&lt;/canConsultate&gt;     &lt;canEndACW&gt;true&lt;/canEndACW&gt;     &lt;canHangup&gt;true&lt;/canHangup&gt;     &lt;canHold&gt;true&lt;/canHold&gt;     &lt;canLeaveConference&gt;true&lt;/canLeaveConference&gt;     &lt;canRetrieve&gt;true&lt;/canRetrieve&gt;");     &lt;canSingleStepTransfer&gt;true&lt;/canSingleStepTransfer&gt;     &lt;canTransfer&gt;true&lt;/canTransfer&gt;     &lt;isOutbound&gt;false&lt;/isOutbound&gt;     &lt;agentOperations&gt;       &lt;agentID&gt;1&lt;/agentID&gt;       &lt;canLogout&gt;false&lt;/canLogout&gt;       &lt;canSetReady&gt;false&lt;/canSetReady&gt;       &lt;canSetNotReady&gt;false&lt;/canSetNotReady&gt;       &lt;canMakeCall&gt;false&lt;/canMakeCall&gt;       &lt;canSetACW&gt;false&lt;/canSetACW&gt;     &lt;/agentOperations&gt;   &lt;/sessionEvent&gt; &lt;/event&gt; </pre>

A classe **AILClient**, cuja instância mantém duas ligações via *socket* para as duas aplicações referidas, foi concebida com vista a flexibilizar o processamento das mensagens recebidas. Neste caso, de processamento de mensagens num formato de XML as classes .net (**XmlSerializer**) dão algumas facilidades na serialização/desserialização de objectos para/de XML.

Implementado o modelo de classes apresentado na **Figura 15**, a obtenção de uma instância do tipo **AILEvent**, que representa um evento de agente ou sessão, é feito por simples desserialização da mensagem recebida do *AIL Agent Server*.

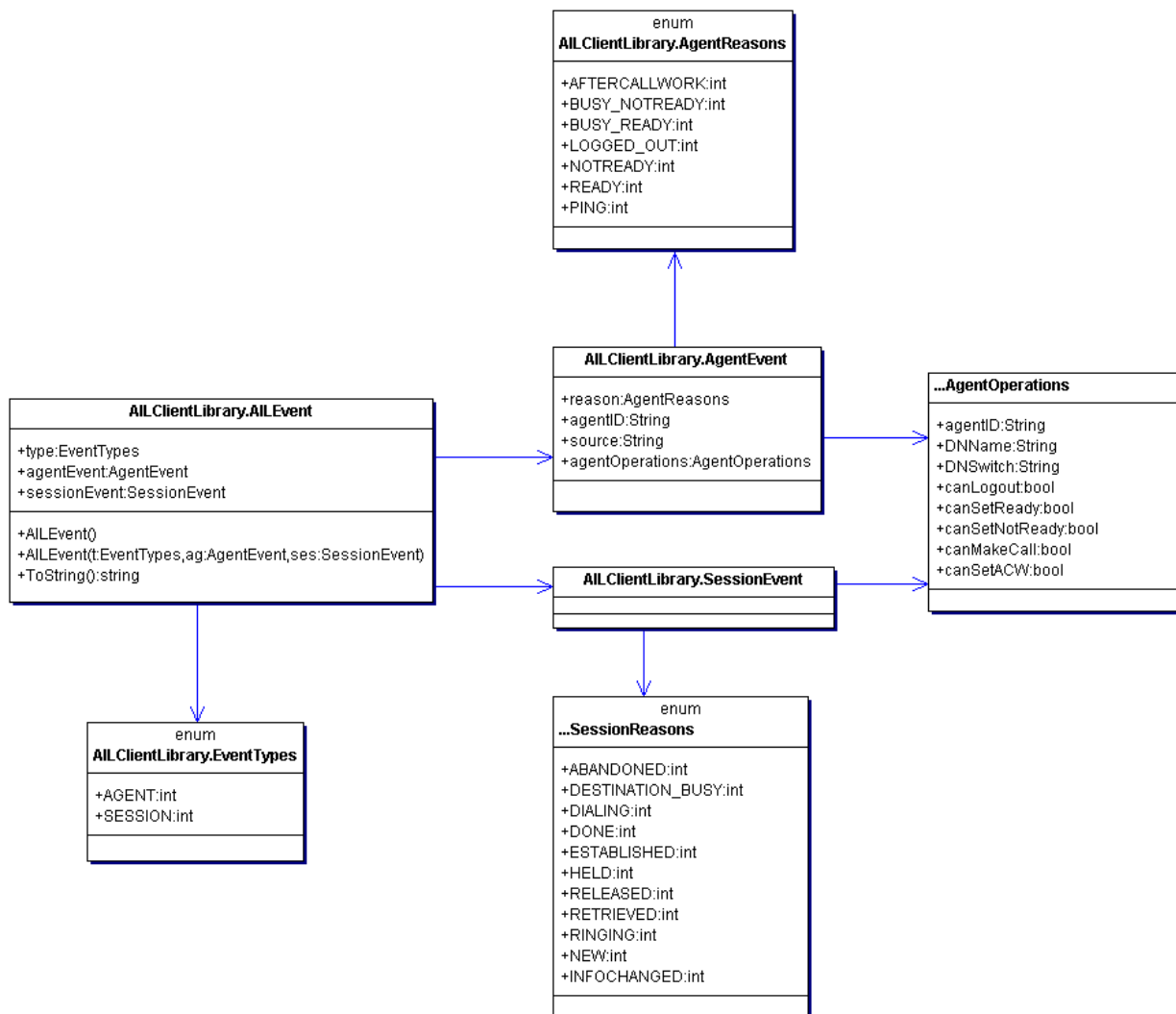
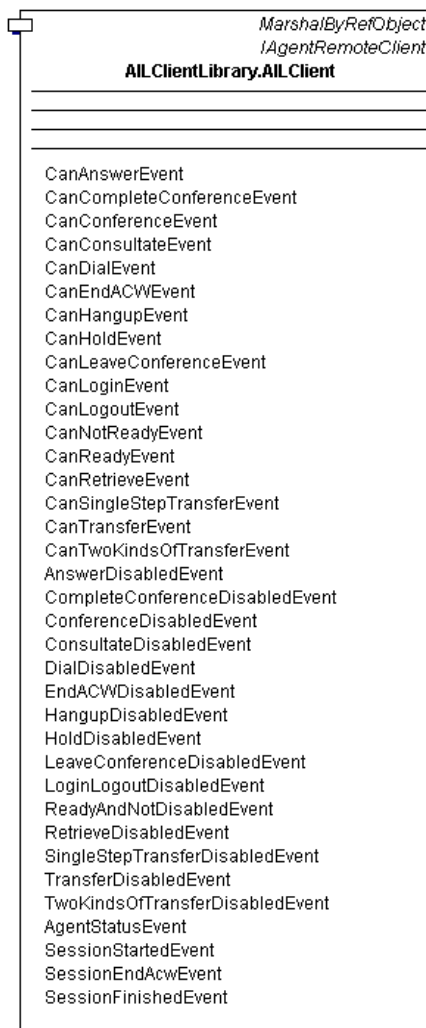


Figura 15 – Diagrama de classes para representação um evento AIL

Com base na informação contida no objecto de **AILEvent**, a instância de **AILClient** irá accionar os respectivos eventos apresentados na Figura 16.



**Figura 16 – Eventos de AILClient**

Além do conjunto de eventos disponíveis na instância de **AILClient**, que mantém actualizado sobre as transições de estado do agente, existem ainda um conjunto de métodos que permitem efectuar operações sobre o agente, através de invocações remotas ao *web service AIL Agent*.

O conjunto operações disponibilizadas está organizado em dois tipos:

- Operações de **Agente** (sobre um **Client ID**);
- Operações de **Sessão** (sobre um **Session ID**).

Isto significa que para a realização de determinadas operações será necessário fornecer um **Client ID** ou um **Session ID**, conforme será visível na descrição de algumas das funcionalidades especificadas.

O **Client ID** é dado pelo *AIL Agent Server* no momento em que instância de **AILClient** se liga a este (ver -o **SocketListener**). Por sua vez o **Session ID** é recebido em determinados eventos do *AIL Agent Server* aquando do estabelecimento de uma ligação telefónica.

Os métodos invocados ao *web service AIL Agent* na realização das operações sobre um agente são:

- **Map** Login(**String** place, **String** username, **String** password, **String** queue)
- **Map** Logout()
- **Map** Login setReady()
- **Map** Login setNotReady()
- **void** MakeCall(**String** destination)
- **void** Hangup(**SessionID** sessionID)
- **void** Hold(**SessionID** sessionID)
- **void** Answer(**SessionID** sessionID)
  
- **Map** getAttachedData(**SessionID** sessionID)
- **String** getAttachedData(**SessionID** sessionID, **String** key)
- **void** setAttachedData(**SessionID** sessionID, **String** key, **String** value)
- **void** deleteAttachedData(**SessionID** sessionID, **String** key)
  
- **Collection** getLoggedAgents()
- **void** Consultation(**SessionID** sessionID, **String** destination)
- **void** Retrieve(**SessionID** sessionID)
- **void** Transfer(**SessionID** sessionID)
- **void** SingleStepTransfer(**SessionID** sessionID, **String** destination)
  
- **void** Conference(**SessionID** sessionID, **String** destination)
- **void** LeaveConference(**SessionID** sessionID)
- **void** CompleteConference(**SessionID** sessionID)
  
- **void** EndSession(**SessionID** sessionID)
  
- **AgentInfo** getAgentInfo()
- **AgentPossibleOperations** getPossibleAgentOperations()
- **SessionPossibleOperations** getPossibleSessionOperations(**SessionID** sessionID)

## ○ CTIToolbar e AILControl

A **CTIToolbar** consiste naquilo que é designado em nomenclatura Windows, como uma **Application Desktop Toolbar**, à semelhança da “vulgar **Windows Taskbar**”. A principal diferença deste tipo de aplicações para uma aplicação Windows é que estas *toolbar's* podem ser ancoradas (**dock**) no *desktop*, reclamando um espaço que não poderá ser sobreposto por mais nenhuma aplicação.

Este tipo de comportamento não é disponibilizado em nenhuma classe da plataforma .net. A única classe que implementa a funcionalidade de uma *toolbar*, `System.Windows.Forms.ToolBar`, apenas serve para ser utilizada dentro de um *Form* não funcionando como uma aplicação separada.

Assim a implementação do comportamento uma *Desktop Toolbar* tem que ser feita directamente sobre a *shell* do Windows. A classe `ApplicationDesktopToolbar` disponibilizada na biblioteca *ShellBasics*, consiste num Windows *Form* com um conjunto de serviços adicionais que permitem manipular a aplicação como uma *Desktop Toolbar*.

O **AILControl** implementa a *user interface* apresentada na **CTIToolbar**. Este controlo foi concebido com o objectivo de que as funcionalidades disponibilizadas pudessem ser facilmente desabilitadas, ou até mesmo retiradas, sem necessitar de recompilar o componente.

Para tal foi tirado partido do mecanismo de configuração da aplicação a partir de um ficheiro XML, *CTIToolbar.config*, disponibilizado pela *framework .net*. Para cada botão apresentado (ver 6 Anexo – Manual de Utilizador da **CTIToolbar**) existe uma entrada no ficheiro *CTIToolbar.config*, com o mesmo nome do membro que detém esse botão, à semelhança do exemplo seguinte:

```
<mButtonLogin
    image = "LOGON.BMP"
    reverseImg = "LOGOFF.BMP"
    tip = "Login"
    reverseTip = "Logout"
    enabled = "true"
/>
```

Cada um dos parâmetros anteriores tem a seguinte especificidade:

- `image` – *Path* para o ficheiro que contém a imagem associada ao botão;
- `reverseImg` - *Path* para a imagem associada à operação inversa do botão;
- `tip` – *Tip* descritiva da operação principal do botão;
- `reverseTip` - *Tip* descritiva da operação inversa do botão;
- `enabled` – *True* ou *false* consoante o botão esteja ou não *enabled* respectivamente.

## 4. Conclusões

O desenvolvimento do **sistema de controlo de Agentes num Contact Center**, teve como principais linhas de orientação:

- Implementar uma estrutura flexível que minimize o esforço de alterações a qualquer um dos níveis do sistema. Este requisito levou a uma repartição sistemática das funcionalidades em bibliotecas e aplicações bem distintas.
- Tirar o maior partido do .net *remoting* na interacção entre a *CTIToolbar* e o *AgentsManager*, minimizando o impacto que a sua utilização poderia ter na implementação e interdependência das aplicações.
- Minimizar o impacto de uma comunicação ao nível do *socket* no desenho do sistema. Neste sentido foi desenhada a classe *SocketListener* numa abordagem *event driven*, em que os seus utilizadores após o lançamento do serviço apenas teriam que subscrever determinados eventos que os notificassem do estado da ligação e das mensagens recebidas.

Além disto, houve que considerar a integração deste sistema num ambiente com outras aplicações, que determinaram alguns dos seus requisitos.

O ambiente de actuação das aplicações do *contact center* é bem confinado a uma rede local (*LAN*) dentro de uma organização. Ou seja, existe todo o controlo da área de administração de redes e segurança sobre as aplicações que podem ser instaladas e executadas neste ambiente (*contact center*).

Neste sentido a concepção do sistema de controlo de Agentes num Contact Center, com principal ênfase na interacção entre a *CTIToolbar* e o *AgentsManager* foi baseado na sua utilização restrita neste ambiente.

Uma possível extensão a este sistema poderia ser o alargamento do acesso do servidor *AgentsManager* a ambientes externos ao *contact center*, com vista a poder ser efectuado um controlo de fora sobre a actividade do *contact center*. Esta extensão obrigaria à inclusão de outros requisitos de validação dos utilizadores e segurança.

Neste tipo de arquitectura é usual que os pedidos de alterações sobre a aplicação *AgentsManager* sejam independentes das alterações da *CTIToolbar*. Ou seja, a entrada em produção de uma nova versão de uma das aplicações não implica a substituição da outra. Como tal, as aplicações devem continuar a funcionar independentemente das actualizações que sejam feitas numa, ou noutra.

Assim foi extremamente importante na concepção das aplicações a definição de uma biblioteca separada com a especificação das APIs que são partilhadas pelas aplicações.

Tendo sido cumpridos os requisitos aqui apresentados o sistema implementado superou alguns dos problemas que são defrontados no desenvolvimento de aplicações distribuídas como a:

- Modularidade – Unidades de funcionamento distintas e confinadas à resolução de um dado objectivo;
- Extensibilidade – Estrutura flexível que minimize o esforço de extensão da aplicação a novas funcionalidades;
- Heterogenidade – Interacção com aplicações em ambientes distintos, como se actuassem no mesmo ambiente.

## 5. Bibliografia

- [1] Microsoft .NET REMOTING – Microsoft Press – Scott McLean, James Naftel e Kim Williams
- [2] Applied Microsoft .Net Framework – Microsoft Press – Richter
- [3] C# and the .NET Platform – Apress - ANDREW TROELSEN
- [4] Accessing Objects in Other Application Domains Using .NET Remoting  
MS.NET FrameworkSDKv1.1
- [5] A Guide to Building Enterprise Applications on the .NET Framework – MSDN Library

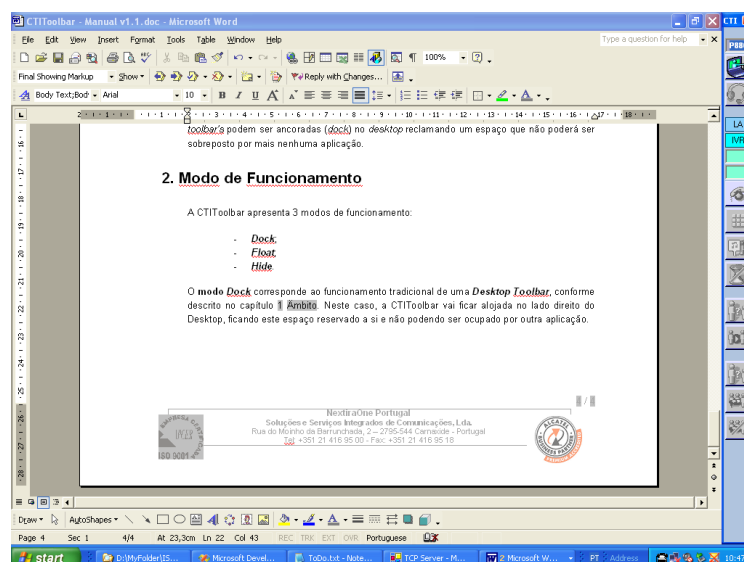
## 6. Anexo – Manual de Utilizador da CTIToolbar

### ○ Modo de Funcionamento

A CTIToolbar apresenta 3 modos de funcionamento:

- **Dock;**
- **Float;**
- **Hide.**

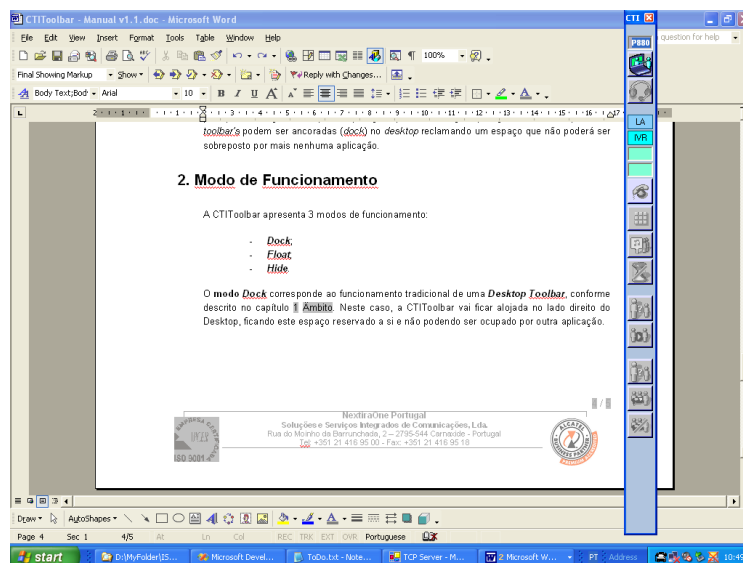
O **modo Dock** corresponde ao funcionamento tradicional de uma **Desktop Toolbar**. Neste caso, a CTIToolbar vai ficar alojada no lado direito do Desktop, ficando este espaço reservado a si e não podendo ser ocupado por outra aplicação.



**Figura 17 – CTIToolbar em modo Dock**

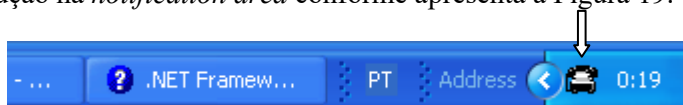
No **modo Float** a CTIToolbar liberta o espaço, que antes lhe estava reservado pelo modo *Dock* podendo este ser ocupado por qualquer outra aplicação. **Este é o modo em que é iniciada a CTIToolbar por defeito.**





**Figura 18 – CTIToolbar em modo *Float***

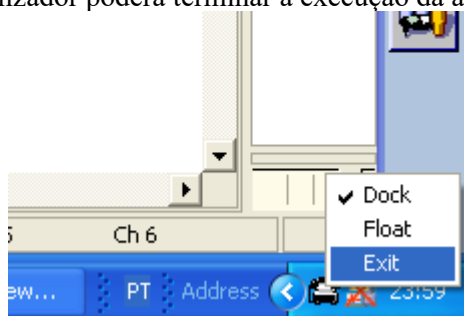
Finalmente, quando o utilizador fecha a **CTIToolbar** esta será ocultada, ficando num **modo *Hide***, sendo apenas visível o seu ícone de execução na *notification area* conforme apresenta a Figura 19.



**Figura 19 – Ícone de execução da CTIToolbar na *notification area***

De notar, que no modo ***Hide*** a **CTIToolbar** continua em funcionamento. Ou seja, se o Agente estiver num estado ***Ready*** (disponível para atendimento) e lhe for entregue uma chamada telefónica a **CTIToolbar** aparecerá automaticamente no desktop, indicando assim a presença da chamada.

Clicando com o botão do lado direito do rato sobre o ícone de execução da CTIToolbar, aparecerá um menu, conforme apresentado na **Figura 20**, que permite alternar o modo de funcionamento da **CTIToolbar** entre *dock* e *float*. É através deste menu que o utilizador poderá terminar a execução da aplicação, na opção ***Exit***.

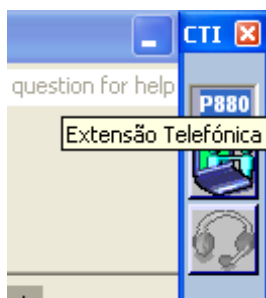


**Figura 20 – Menu da CTIToolbar**

## ○ Funcionalidades

O desenho da **CTIToolbar** teve em vista minimizar o espaço ocupado por esta aplicação sobre o *desktop* do Agente. Seguindo esta abordagem uma das opções tomadas foi omitir as legendas de todos os botões e campos de texto.

Em alternativa, foi implementada uma *ToolTip* para cada um dos campos da **CTIToolbar** que indica qual a funcionalidade disponibilizada. Esta *ToolTip* “salta” automaticamente quando é deixado o rato sobre o respectivo campo, conforme apresenta a **Figura 21**.



**Figura 21 – Exemplo de uma ToolTip: campo de texto para a extensão telefónica**



Figura 22 – Layout da CTIToolbar e respectivas funcionalidades

## ■ Login e Logout

No momento em que é iniciada a **CTIToolbar** o único botão que se encontra disponível é o botão de Login. Para a concretização desta operação deverão ser introduzidos os seguintes dados:

- **Extensão Telefónica** – número da extensão com um “P” (maiúsculo) em prefixo;

- **Username, Password e Queue** – Estes campos são introduzidos através de um *form* específico, que é apresentado após o “click” sobre o botão de Login.

**Figura 23 – Form para introdução do Username, Password e Queue**

Se o Login for concretizado com sucesso o mesmo botão passará a disponibilizar a funcionalidade de Logout e vice-versa, conforme apresenta a **Figura 24**.



**Figura 24 – Botão de Login e Logout**

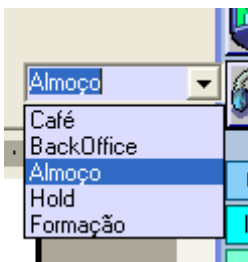
#### ▪ **Ready e NotReady**

Se a operação de Login for concretizada com sucesso o agente passa a ter a possibilidade de alternar entre os estados Ready (disponível para atendimento) e NotReady (ou **Busy**) através do botão apresentado na **Figura 25**.



**Figura 25 – Botão de Ready e NotReady**

Para concretizar a operação de NotReady o Agente terá de indicar uma razão (*NotReady Reason*) dentro de uma lista de opções apresentada.



**Figura 26 – NotReady Reasons**

### ▪ Estado do Agente

Por baixo do botão de Ready/Not Ready aparecerá uma legenda que indicará o estado do agente, que transitará entre: **Busy**, **Ready** e **ACW** (*after call work*).



**Figura 27 – Estados do Agente**

### ▪ Customer Center

Esta operação serve para enviar à aplicação **Customer Center** o identificador do Cliente/Número de origem da chamada.

Caso o Customer Center não esteja disponível será apresentada uma mensagem notificando o agente de que esta aplicação não se encontra disponível, interrogando-o se pretende tentar enviar novamente a informação ou cancelar.



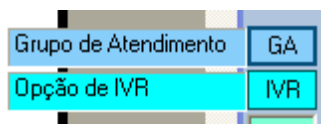
**Figura 28 – Customer Center**

### ▪ Linha de Atendimento e Opção no IVR

Quando é entregue uma chamada de um cliente ao Agente poderão existir dados associados a essa chamada que indicam:

- **Linha de Atendimento** – Grupo de entrada da chamada no *contact center*.
- **Opção no IVR** – Opção seleccionada pelo Cliente no menu do IVR.

Estas informações serão apresentadas no momento em que a chamada chega ao Agente através de um *PopUpField*, que desaparecerá ao fim de 3 segundos. Para o Agente rever a informação destes campos bastará clicar sobre os respectivos botões.



**Figura 29 – Linha de Atendimento e Opção no IVR**

### ▪ **Tempo da chamada em espera**

Outro dos dados que poderá ser dada juntamente com a chamada será o tempo que esta esteve em espera até ser entregue ao Agente. Esta informação estará sempre visível na CTIToolbar como apresenta a **Figura 30 – Tempo de espera e duração da chamada**.

### ▪ **Duração da chamada**

Tempo de duração da sessão telefónica desde que a chamada foi entregue ao Agente. Este tempo terminará quando o Agente seleccionar o Fim de Sessão (0) ou se entrar uma nova chamada.



**Figura 30 – Tempo de espera e duração da chamada**

### ▪ **Atender e Desligar**

A **Figura 31** apresenta o estado do botão atender e desligar quando disponibiliza cada uma das respectivas operações.



**Figura 31 – Atender e Desligar**

**NOTA:** No *contact center* da Radiomóvel, estando por defeito a CTIToolbar configurada para **atendimento automático**, ao receber uma nova chamada este botão disponibilizará de imediato a funcionalidade desligar.

### ▪ **Ligar**

Ao ser seleccionada esta operação é apresentada uma caixa de texto onde o utilizador deverá introduzir o número de destino. Se por premido “ENTER” a operação é concretizada. Para cancelar basta premir o “ESCAPE” ou fechar a janela no *icon* de *close*.



Figura 32 – Estabelecimento de uma chamada

## ▪ Chamada em Espera (*Hold*) e Recuperação

A

Figura 33 apresenta os dois estados deste botão para a operação de Hold (colocar chamada em espera) e Recuperação, respectivamente.



Figura 33 – Chamada em Espera (*Hold*) e Recuperação

## ▪ Fim de Sessão

Quando uma chamada telefónica é finalizada o agente entrará em período de **Wrap Up**, durante o qual não receberá outras chamadas telefónicas vindas do exterior. No entanto, caso o agente entenda, poderá antecipar o fim deste período seleccionando a operação de Fim de Sessão.



Figura 34 – Fim da Sessão

## ▪ Consulta

Apresenta uma nova janela com a lista de agentes disponíveis para receber a transferência de uma chamada. Em alternativa, o agente poderá introduzir um número de destino, para onde pretende fazer a consulta.

Quando esta operação é concretizada a chamada com o cliente fica em espera e é feita uma nova ligação a outro agente.

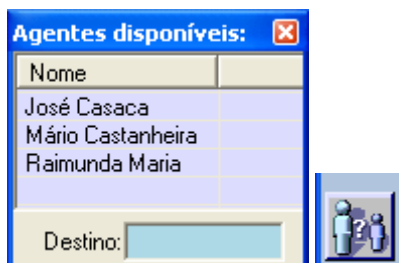


Figura 35 – Consulta e apresentação dos agentes disponíveis

## ▪ **Transferência**

Completa a transferência de uma chamada do cliente para um outro agente que foi contactado através da operação de Consulta (ver 0

**Consulta**).

Se não tiver havido uma consulta prévia então é apresentado uma janela semelhante ao da operação de Consulta e a chamada será transferida de uma só vez para o destino.



**Figura 36 – Transferência**

## ▪ **Consulta para Conferência**

Semelhante à operação de Consulta, mas com o objectivo final de fazer uma conferência com outro agente e o cliente.

Quando esta operação é concretizada a chamada com o cliente fica em espera e é feita uma nova ligação a outro agente.



**Figura 37 – Consulta para Conferência**

## ▪ **Estabelecimento de Conferência**

Faz a conferência da chamada com um agente que está em consulta para conferência, mais o Cliente que ficou em espera.



**Figura 38 – Estabelecimento de Conferência**

## ▪ **Abandono de Conferência**

Após o estabelecimento de uma conferência telefónica esta operação deixará o Cliente a falar com o segundo Agente libertando o primeiro da chamada.

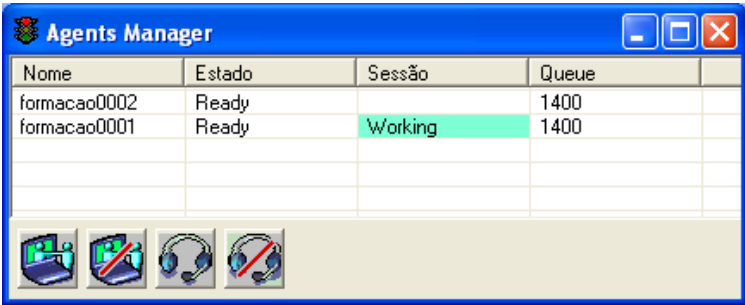


**Figura 39 – Abandono de Conferência**



## - Anexo – Manual de Utilizador do *AgentsManager*

O *AgentsManager* é uma aplicação que permite controlar o estado dos agentes que estejam ligados à *CTIToolbar*, no *contact center*.



Nome	Estado	Sessão	Queue
formacao0002	Ready		1400
formacao0001	Ready	Working	1400

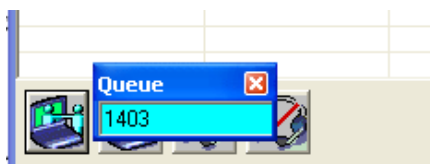
Figura 40 – Layout do *AgentsManager*

Quando um agente efectua a operação de **Login** na *CTIToolbar*, automaticamente é apresentada uma nova entrada no *AgentsManager* para o respectivo agente, com a seguinte informação:

- Nome – *Username* com que o agente efectuou o *login*;
- Estado – Estado do agente (Busy, Ready e ACW). Ausência de estado, significa que o agente mantém em execução a *CTIToolbar* mas saiu da linha de atendimento (Loggof).
- Sessão – Indica se o agente tem uma chamada em curso. Quando tiver finalizado o período de ACW se o agente não tiver finalizado a sessão, esta passará a cor vermelha.
- Linha de atendimento a que o agente está ligado.

Sobre um determinado agente é possível efectuar as seguintes operações:

- Login – Liga o agente a uma determinada linha de atendimento dada pelo supervisor, através de um *form* específico conforme apresenta a figura seguinte:



- Logout – Retira o agente da linha de atendimento a que este se encontra ligado.
- Ready – Põe o agente disponível para efectuar e receber chamadas na linha de atendimento a que está ligado.

- NotReady – Põe o agente indisponível para efectuar ou receber chamadas na linha de atendimento a que está ligado.