ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

"Diseño e implementación de una aplicación CTI basada en el estándar TAPI para el control de centralitas de conmutación telefónica"

AUTOR: Luis Miguel Rodriguez Cegarra
DIRECTOR: Pablo Pavón Mariño

Enero / 05

Autor	Luis Miguel Rodriguez Cegarra
E-mail del Autor	Lm.rodriguez@ya.com
Director(es)	Pablo Pavón Mariño
E-mail del Director	pablo.pavon@upct.es
Codirector(es)	
Título del PFC	"Diseño e implementación de una aplicación CTI basada en el estándar TAPI para el control de centralitas de conmutación telefónica"
Descriptores	

Resumen

Se ha desarrollado una aplicación para el sistema operativo Windows basado en el estándar TAPI de Microsoft que permite interoperar con dispositivos telefónicos, principalmente centralitas digitales, y que implementa funciones de control de llamadas, control de contenidos y generación de estadísticas del tráfico.

Titulación	Ingeniería de Telecomunicación, especialidad en Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Enero - 2005

Índice general

Capítulo 0: Introducción……7

0.1 Motivación ……9
0.2 Objetivos ……9
0.3 Estado del arte ……10
0.4 Organización del proyecto ……10

Capítulo 1: La tecnología CTI ……13

1.1 Un poco de historia……13
1.2 Aplicaciones de la tecnología CTI ……15
1.2.1 Control de gasto: tarificación ……15
1.2.2 Control de llamadas ……17
1.2.3 Control de contenidos ……19
1.2.4 Enlace con base de datos ……20
1.2.5 CRM (Customer Relationship Management) ……..21
1.3 Dispositivos telefónicos ……23
1.4 Arquitecturas Hardware para los sistemas CTI ……..25
1.4.1 Acceso directo o “first-party” ……25
1.4.2 Diferentes esquemas para el acceso directo ……..27
1.4.3 Acceso indirecto o “third-party” ……..29
1.4.4 Diferentes accesos para el acceso indirecto ……..31
1.5 Arquitecturas Software ……32
1.5.1 Capa única ……..32
1.5.2 El modelo cliente/servidor en CTI ……33

Capítulo 2: El estándar TAPI y el protocolo CSTA……..35

2.1 Introducción ……35
2.2 Arquitectura TAPI: Windows Open Services Architecture……36
2.3 TAPI 1.x ……..38
2.4 TAPI 2.0 ……..39
2.5 TAPI 2.1 ……..40
2.6 TAPI 3.x ……..41
2.7 Estándar de conexión telefonía ordenador. Protocolo CSTA……..45

Capítulo 3: Manual de usuario de PBX Control Center…...48

3.1 Bienvenido a PBX Control Center ……48
3.2 Requisítos mínimos ……49
3.3 Ventana principal……50
3.4 Vista ‘Monitorización’……..54
3.5 Vista ‘Base de datos de llamadas’……58
3.6 Vista ‘Configuración’……64
3.7 Vista ‘Visor de Informes’……68
3.8 Generación de informes estadísticos……69
3.9 Formato del fichero de configuración……73

Capítulo 4: Desarrollo de la herramienta……78

4.1 Introducción……..78
4.2 Objetivos iniciales……78
4.3 Estándar CTI utilizado……80
4.4 Entorno de desarrollo……..80
4.5 Librerías y herramientas utilizadas……82
4.6 Algunos aspectos de diseño……..85
4.6.1 Acceso a base de datos……85
4.6.2 Generación de informes……87
4.6.3 Tarificación……..88
4.6.4 Ayuda de la aplicación……..89
4.7 Escenario de las pruebas……...89

Capítulo 5: Conclusiones y líneas futuras……..90

5.1 Conclusiones……90
5.2 Líneas futuras……..91

Apéndice A: Configuración de un origen de datos ODBC.93
Apéndice B: Tarificación en el estándar TAPI……...95
Apéndice C: Creación de ayuda en formato CHM……..103
Referencias……104

Capítulo 0
Introducción

0.1 Motivación

La evolución de las telecomunicaciones y el abaratamiento de las centralitas digitales o PBX ha hecho que cualquier mediana o pequeña empresa pueda disponer en su infraestructura de una centralita digital privada. Históricamente, el único software existente para administrar estas centralitas ha sido el software de configuración desarrollado por el fabricante y adjuntado con la propia PBX. Este software es válido únicamente para un determinado modelo de centralita y la comunicación entre el PC y la centralita hace uso de protocolos propietarios y cerrados. A parte de este software, hoy en día, no existen muchas compañías que desarrollen aplicaciones para centralitas (excepto software de tarificación, que también está ligado a ciertos modelos de PBX), ya que normalmente estas aplicaciones deben ser concebidas para un modelo o marca específica de PBX siendo incompatible para el resto. Esta razón reduce considerablemente el nicho de mercado.

El auge de la tecnología CTI *(Computer Telephony Integration)* y el desarrollo y popularización de estándares CTI, como el estándar TAPI *(Telephony Application Program Interface)* [1], permiten integrar las ventajas de los dispositivos telefónicos y del PC haciendo posible desarrollar aplicaciones para centralitas u otros dispositivos independientemente de la marca o modelo. Esta independencia del hardware amplía de nuevo el mercado a todas las centralitas que implementen dicho estándar que, actualmente, son la mayoría.

En este contexto, en el presente proyecto se desarrolla una aplicación CTI basada en el estándar TAPI con funcionalidad de control y monitorización de llamadas entre otras.

0.2 Objetivos

Una vez situados en el marco tecnológico, el objetivo del proyecto actual es desarrollar una herramienta software CTI basada en el estándar TAPI de Microsoft [2] para el sistema operativo Windows, capaz de interactuar con todas las centralitas privadas digitales que implementen esta especificación y de las que se disponga del driver TAPI, normalmente proporcionado por el fabricante.

La herramienta debe ser capaz de monitorizar el tráfico de llamadas en tiempo real de la centralita permitiendo algún tipo de control de las llamadas (bloqueo de extensiones, colgar llamadas, número prohibidos, etc) así como mantener un registro del tráfico para permitir posteriormente un análisis off-line.

Se exige además que la herramienta disponga de un interfaz amigable e intuitivo que conlleve asociado todas las características de un programa típico en el Sistema Operativo Windows.

0.3 Estado del arte

A pesar de ser CTI una tecnología que permite desarrollar aplicaciones que mejoran la productividad y ahorro de costes, son muy pocas las empresas en España dedicadas al desarrollo de este tipo de software. Actualmente encontramos principalmente la empresa **cHar** con sede en Barcelona (www.char.es) especialista en soluciones CTI para empresas, hoteles y aplicaciones soft-phone. También se encuentra con sede en la ciudad de Vigo la empresa **Seintel** (www.seintel.com) que ha desarrollado una familia de aplicaciones CTI de tarificación llamada CTIWare aunque al parecer abandonó el desarrollo. A partir de ahí, existen varias empresas localizadas en varias ciudades españolas aunque sin alcanzar la popularidad o calidad de software que, por ejemplo, cHar.

0.4 Organización del proyecto

El presente texto consta de un total de 6 capítulos distribuidos por su distinta temática en los siguientes puntos:

- **Capítulo 0:** Introducción
- **Capítulo 1:** La tecnología CTI
- **Capítulo 2:** El estándar TAPI y el protocolo CSTA
- **Capítulo 3:** Manual de usuario de PBX Control Center
- **Capítulo 4:** Desarrollo de la herramienta
- **Capítulo 5:** Conclusiones y líneas futuras

El primero de los capítulos es de **introducción** donde se trata en términos generales qué se va a hacer, por qué, es estado del arte y el propio contenido del texto.

El segundo de los capítulos trata sobre **la tecnología CTI** echando una mirada a sus orígenes y presentando cuáles son las posibles configuraciones que se pueden adoptar.

El tercero de los capítulos se centra en el estándar de telefonía **TAPI** desarrollado por Microsoft ya que ha sido el estándar utilizado en el desarrollo de la herramienta. Finalmente presenta el protocolo CSTA que utiliza el estándar TAPI en la comunicación entre PC y dispositivo telefónico.

En el capítulo 3 se incluye el **manual de usuario** de la herramienta desarrollada: PBX Control Center.

El penúltimo capítulo abarca las herramientas y medios utilizados para el **desarrollo de la herramienta** justificando las decisiones tomadas en el diseño y presentando el hardware utilizado en el escenario de las pruebas.

Finalmente, se engloba en el último de los capítulos las **conclusiones y líneas futuras** de investigación que se pueden seguir tras el presente proyecto.

Capítulo 1
La tecnología CTI

1.1 Un poco de historia

Actualmente, en España, la tecnología CTI está mereciendo cada vez mayor interés en el panorama económico y empresarial. La principal causa de esto, podemos encontrarla en el auge de las telecomunicaciones, donde hay cada vez mayor número de operadores y servicios, y en el de la informática, ya que estamos hablando de sistemas que permiten integrar los beneficios de ambas tecnologías obteniendo un valor añadido superior.

Las siglas CTI son un acrónimo anglosajón que significa "Computer Telephony Integration". La tecnología CTI surge de la idea de combinar las ventajas que se obtienen de los sistemas telefónicos y de los informáticos. Cuando llamamos a un servicio de atención al cliente y nos responde un sistema automático que nos guía a través de menús estamos interactuando con un sistema CTI; éste es un sistema que no se limita a controlar llamadas (algo que también hace, ya que es capaz de responder, colgar, transferir…) sino que también puede introducir información en la línea telefónica (en forma de voz, como en este caso, datos, fax…) y extraer información de ella (en forma de tonos multifrecuencia –DTMF- o, incluso, por reconocimiento de voz).

CTI tiene sus orígenes a principios de los 80. AT&T fue el primer fabricante de PBX en adjuntar a su 'Dimension PBX' aplicaciones que proporcionaban funciones de directorio y centro de mensajes. Las llamadas no contestadas eran redireccionadas a un agente de control de mensajes el cual se comunicaba con un servicio de directorio para buscar la persona a quien iba dirigida la llamada. Entonces, el mensaje se encaminaba a una impresora próxima a ésta. La conexión entre la PBX y esta aplicación era un simple enlace físico que no proporcionaba funciones avanzadas de voz.

El concepto de mejorar las funciones de la PBX a través de una aplicación adjunta, con conexiones físicas y lógicas entre éstos dos sistemas independientes, no fue llevado a cabo hasta finales de los 80. Los primeros sistemas CTI "verdaderos" surgieron casi simultáneamente en dos sistemas PBX. La *NEC NEAX2400* y la *Intecom IBX* ofrecían una especie de API que permitía a desarrolladores independientes diseñar aplicaciones que se ejecutarían en sistemas cliente o servidor independientes para mejorar las capacidades de la PBX. Ambas opciones usaban un sistema inteligente de señalización entre la PBX y la aplicación. Tal sistema permitía a la PBX enviar mensajes de estado a la aplicación y proporcionaba los mecanismos para que la aplicación enviara comandos a la PBX. Estas "APIs" o herramientas de desarrollo de software incluidas en estos sistemas eran propietarias y obligaba a los desarrolladores de software a escribir programas diferentes para cada sistema. Esta situación hizo que los primeros estándares CTI surgieran a finales de los 80 y principios de los 90. Fueron las mayores empresas de computadores, como DEC y IBM, las que tomaron la iniciativa, no los fabricantes de PBX.

Las aplicaciones cliente/servidor CTI fueron inicialmente conducidas por Novell, quien promovió su estándar *Telephony Services Application Programming Interface* (TSAPI) [3]. En Europa, el estándar CTI fue desarrollado por el *European Computer Manufacturers Association (ECMA)* [4] y es conocido como *Computer Services Telephony Applications (CSTA)*. Microsoft desarrolló su propio estándar CTI, *Telephony Applications Programming Interface (TAPI)*, que posteriormente fue mejorado para soportar aplicaciones cliente/servidor sustituyendo al TSAPI de Novell como la plataforma más popular de desarrollo CTI.

A partir de hay surgen diferentes arquitecturas de conexión entre el dispositivo telefónico y el computador así como diferentes APIs de programación, analizadas en secciones posteriores.

1.2 Aplicaciones de la tecnología CTI

1.2.1 Control de gasto: tarificación

Puede decirse que la tarificación es la aplicación CTI más antigua que existe. La tarificación consiste en obtener información de todas las llamadas realizadas y asignarle de alguna forma un precio a cada una de ellas.

Las primeras aplicaciones de tarificación fueron usadas por los operadores para emitir las facturas correspondientes al consumo de los usuarios. Por tanto, podemos decir que la tarificación existe desde que existe la telefonía y que seguirá existiendo siempre. La introducción de los ordenadores en los procesos de tarificación de los operadores ha permitido la creación de planes de precios muy variados y específicos que les permiten mejorar su oferta de servicios.

Hoy en día se han desarrollado aplicaciones de tarificación más avanzadas que el mero generador de informes de coste:

- Retarificadores: los retarificadores se usan en combinación con un tarificador clásico. Se trata de comparar operadores aprovechando que se tienen almacenadas las llamadas de periodos significativos. Esto es: si con un operador A, hemos gastado X dinero el mes pasado, podemos usar la información que almacenó el tarificador (destino y duración de todas las llamadas) para comprobar "cuál hubiera sido el gasto con el operador B".
- Enrutado inteligente en tiempo real: todas las centralitas son capaces de realizar algún tipo de enrutamiento de llamadas. Esto significa que, en función del número de destino se eligen diferentes operadores con el objeto de abaratar los costes. La elección de operador se puede hacer realizando la llamada por una línea exterior que esté preseleccionada para ese operador, o bien, mediante el marcaje de prefijos de acceso indirecto (los prefijos son marcados automáticamente por la centralita). El enrutado programable en las centralitas tiene algunas limitaciones, por ejemplo:

puede no tener en cuenta que las tarifas cambian con las franjas horarias o que cambian en días festivos. Por eso, es más eficaz tener una aplicación informática de control de enrutado. Lo ideal sería tener un ordenador que calcule en cada momento el operador óptimo para cada llamada. Realizar ese cálculo en tiempo real es poco viable por el gran volumen de operaciones a realizar, por lo que estas aplicaciones calculan las tablas de encaminamiento, que únicamente son recalculadas, cuando hay cambios de tarifas.

Los aspectos técnicos generales de una aplicación de tarificación son los siguientes:

- Enlace de datos: es necesario establecer un enlace entre el ordenador que registrará las llamadas y la centralita. Casi todos los modelos de centralitas (Ericsson, Bosh…) utilizan un puerto serie para esta labor. En el momento que finaliza la llamada, la centralita envía al ordenador una trama con la información de tarificación. El formato de las tramas suele diferir de un fabricante a otro pero suelen ser tramas ASCII de fácil interpretación (si están documentadas). A diferencia de otras aplicaciones CTI donde existen estándares (TAPI, TSAPI…) en la tarificación cada fabricante va por su lado. En este caso es el programador el que tiene que hacer su aplicación lo bastante versátil para admitir diferentes formatos de trama.

- Cálculo de coste: el cálculo suele hacerse utilizando la información de tarificación que da la central. Esta información suele ser: número origen, número destino, línea de salida (puede ser exclusiva para un operador o no), prefijo de operador (si está en blanco, el operador usado será el preseleccionado en la línea), fecha y hora de inicio y fecha y hora de fin. El cálculo se basa en el manejo de bases de datos de operadores y tarifas; suele ser un proceso complejo. A veces, se utilizan los pasos de tarificación proporcionados por los operadores.

El servicio de pasos consiste en que el operador envía una señal inaudible cada vez que se supera una cierta unidad de tarificación (paso). Para saber el precio hay que tener el servicio activado, tener un contador de pasos y multiplicar el número de pasos final por el precio del paso (eso en una línea individual, sistema muy utilizado en teléfonos públicos). Las centrales son capaces de contar los pasos y de incluir el número en las tramas de tarificación, con lo que el cálculo de coste es trivial. Sin embargo, este servicio se usa poco porque no todos los operadores lo proporcionan y los que lo hacen cobran por su uso.

1.2.2 Control de llamadas

El control de llamadas es la capacidad más importante de los sistemas CTI. De hecho, aunque los tarificadores privados son antiguos, la gente no empezó a hablar de aplicaciones CTI hasta que aparecieron las primeras aplicaciones de control de llamadas, a principios de 1999. Esto quizás se deba al efecto psicológico de no ver ninguna integración hasta que se pueden efectuar operaciones telefónicas básicas (llamadas, transferencias, conferencias…) desde el ordenador.

El control de llamadas consiste por tanto en la capacidad de realizar operaciones telefónicas desde un programa informático y también en ser capaz de obtener desde el programa toda la información posible (toda la que tengan nuestros dispositivos telefónicos) sobre las llamadas en curso en el sistema (eso incluye las generadas desde el ordenador y las generadas manualmente por los usuarios al, por ejemplo, descolgar el teléfono).

Existen diversos medios para programar aplicaciones de este tipo, los más importantes son los estándares TAPI y TSAPI que veremos posteriormente. Independientemente del estándar que usemos, desde el punto de vista del programador, el control de llamadas tiene dos conceptos básicos:

- Una serie de funciones que permiten realizar acciones sobre las llamadas: marcar (crear una llamada), responder, colgar, transferir, iniciar conferencia, retener, liberar, capturar llamada…
- Un método para que los dispositivos telefónicos nos informen de los eventos que ocurren en ellos: llamada entrante, llamada saliente, llamada colgada…

Nótese que cuando hacemos una acción telefónica desde programa vamos a provocar eventos. Nótese también que podemos tener conocimiento de una llamada provocada por agentes externos al programa (por ejemplo, una llamada entrante desde la red pública) a través de eventos. Una vez que recibimos el evento que nos anuncia una llamada (aunque no la provocáramos nosotros) podremos realizar acciones sobre ella, por ejemplo: las llamadas externas las podemos contestar y después colgarlas. Cuando generamos una llamada desde programa, por ejemplo, con la función "marcar" (que según el estándar tendrá diferentes nombres) la llamada se genera en un teléfono físico y activa el dispositivo de manos libres. Si no existe manos libres, la llamada no es efectiva hasta que un usuario descuelgue físicamente el teléfono. Nótese que aunque la llamada se haya generado desde el ordenador va a estar en manos de un usuario que puede actuar manualmente sobre ella: puede colgarla, retenerla…

Con todo el párrafo anterior se quiere dejar claro lo siguiente: "en CTI se puede actuar sobre una llamada en cualquier momento, tanto desde programa como manualmente". Para cada llamada el sistema (el estándar CTI) suele proporcionar una serie de datos que buscando el conjunto común a diferentes estándares serían:

- Identificador: es un número único para la llamada. Ese identificador hay que usarlo en cualquier función que vaya a modificar la llamada. Se obtiene a través de eventos, salvo cuando ejecutamos una función que crea

una nueva llamada, en este caso la propia función devuelve el identificador.

- Número llamante: es muy interesante obtenerlo en las llamadas que vienen del exterior.

- Número llamado.

- Estado: una llamada puede tener varios estados, a saber: "sonando" (la llamada ha entrado pero no ha sido contestada), "hablando", "retenida" y "finalizada" son los más importantes. Por ejemplo, el estándar TAPI distingue más estados: cuando se realiza una llamada al exterior, antes de que llegue a sonar al otro lado pasará por un estado de "en camino", también es posible que nunca llegue a sonar porque ha alcanzado el estado de "línea ocupada" o el de "destino no existe".

1.2.3 Control de contenidos

En el apartado anterior hablamos de cómo controlar llamadas (*call control*) pero en ningún momento hablamos de la información que fluye a través de ella. Algunos estándares como TAPI (no así TSAPI) abren la puerta al control de contenidos (media control).

Ejemplos de control de contenidos serían los siguientes:

- Enviar/recibir datos a través de un módem

- Utilizar un módem de voz (o una tarjeta de sonido conectada a un módem) para grabar o reproducir mensajes. Usando estas técnicas se pueden crear aplicaciones de mensajería vocal. Un aspecto interesante es la integración con módulos de síntesis y/o reconocimiento de voz.

- Reconocer señales enviadas por el usuario al otro lado de la línea, fundamentalmente los tonos generados por los teléfonos multifrecuencia (tonos DTMF). Esta es la base para crear portales telefónicos.

- Utilizar un módem/fax (o una tarjeta específica) para enviar y/o recibir faxes. Combinando la capacidad de enviar fax con la de reconocer tonos podemos crear el fax bajo demanda.

Son especialmente interesantes las aplicaciones de mensajería integrada. Estas aplicaciones permiten convertir un fax entrante en un correo electrónico. El fax se puede incluir como una imagen adjunto (caso más fácil) o se puede convertir en texto usando una aplicación O.C.R. (*Optical Character Recognition*). Sólo en el caso de tener el O.C.R. podremos saber quién es el verdadero destinatario del fax para enviarle el correo electrónico. Si no hay O.C.R. tiene que haber un encargado humano de recibir y distribuir estos correos. Una aplicación de este tipo también permite convertir correos en faxes, por ejemplo, enviando un e-mail a la dirección fax@dominio-interno y poniendo en el subject el número de destino. El software deberá generar el archivo de datos para el fax, que en el fondo es un formato de imagen binivel con compresión sin pérdidas, hay diferentes estándares: grupo III, grupo IV…).

Las aplicaciones de mensajería integrada también pueden integrar la mensajería vocal. De este modo pueden convertir los mensajes de voz en correos electrónicos. Igual que con el fax, la solución trivial es adjuntar un fichero de voz, la más compleja consiste en reconocer la voz. También existen sistemas que permiten la consulta del correo electrónico llamando a un número de teléfono (el usuario se autentica con una clave de dígitos DTMF y un sintetizador de voz "le lee" su correo).

Microsoft ha apostado por hacerle fácil a los desarrolladores la creación de estas aplicaciones. El método elegido ha sido integrar TAPI con otras API's como la MAPI (Multimedia API) que permite grabar y reproducir voz y la SAPI (*Speech API*) que implementa funciones de reconocimiento y síntesis de voz.

1.2.4 Enlace con bases de datos

Casi todas las aplicaciones CTI utilizan la información obtenida del sistema telefónico para realizar búsquedas en bases de datos. Otras veces es al revés: la información presente en las bases de datos se usa para utilizarla en el sistema telefónico. De hecho, el programador CTI suele trabajar más tiempo con el acceso a bases de datos que en la programación de dispositivos telefónicos (en

los equipos humanos grandes, la parte del equipo dedicada a bases de datos suele ser más numerosa que la dedicada a la parte CTI pura):

Como ejemplos prácticos del uso de bases de datos en sistemas CTI tenemos:

- La propia tarificación es un enlace de la información de las llamadas (muchas veces almacenada en bases de datos de llamadas) con bases de datos de operadores y tarifas.
- Para que la tarificación resulte más interesante hay que adjuntar bases de datos administrativas (división en departamentos, extensiones, nombres de empleados…). Sólo así se pueden generar informes detallados por unidades administrativas, usuarios individuales…
- Son muy interesantes los sistemas que obtienen el número origen de las llamadas entrantes y hacen algún tipo de búsqueda con él. Por ejemplo: buscarlo en la base de datos de clientes y abrir su ficha automáticamente, en caso de que el llamante no sea cliente se puede abrir una ficha en blanco con el campo de teléfono ya cubierto. Otro ejemplo típico son los sistemas de apoyo a los servicios de emergencia, con la base de datos adecuada se puede localizar al llamante en el mapa de la ciudad.
- Por último, otra aplicación típica es la realización de campañas telefónicas. Por ejemplo, se pueden buscar las fichas de los clientes cuyo apartado de “saldo pendiente” sea positivo (es decir: que tengan facturas sin pagar) y el ordenador puede elaborar una lista de teléfonos que puede ir marcando automáticamente. Otro ejemplo sería buscar a todos los que compraron determinado producto y llamarlos para ofrecerles una nueva versión o una opción a mayores.

1.2.5 CRM (Customer Relationship Management)

CRM es un acrónimo anglosajón, que hace relación a la gestión de la atención al cliente. Muchas empresas están hoy día invirtiendo en sistemas CRM lo que hace que otras muchas se dediquen a desarrollarlos y comercializarlos. Cuando hablamos de sistemas CRM debemos pensar en todas las formas en las

que un cliente puede interactuar con una empresa: correo electrónico, aplicaciones Web, envío/recepción de fax y, como no, voz telefónica. Podemos deducir que las técnicas CTI son muy importantes para el CRM, aunque sólo son una parte del conjunto. Las tendencias actuales tratan de conseguir sistemas integrados, esto es: que desde una página Web se pueda solicitar el envío de un fax o la llamada telefónica de un experto.

La aplicación, que antes comentábamos, que es capaz de abrir la ficha de un cliente cuando éste llama (buscándolo por su número de teléfono) es una aplicación típica CRM. Otra aplicación muy extendida son los llamados argumentarios. Un argumentario es una ayuda automática a los teleoperadores. Se trata de guiar al operador por los escenarios típicos en los que va a desarrollarse la consulta de un cliente. El sistema informático debe abrir el argumentario automáticamente antes de que el agente responda la llamada. Es típico que los agentes de un *call center* puedan llevar consultas de varios tipos (el mismo agente puede responder a temas de tipo comercial y técnico), en estos casos se le debe asignar un número de teléfono de entrada diferente a cada servicio y el sistema debe abrir el argumentario que corresponda. Otro uso de los argumentarios es en las campañas telefónicas donde el agente es el que llama (muchas veces, ayudado por el sistema CTI) y el argumentario le guía; si lo que se realiza en cada llamada es una encuesta el agente puede ir anotando las respuestas que quedarán registradas en una base de datos.

Las centralitas telefónicas de cierto tamaño suelen incluir una característica importante para ayudar a las aplicaciones CRM. Se trata de los grupos ACD (Automatic Call Distribution). En una centralita se pueden definir varios grupos ACD a la vez. Cada grupo consiste en un número telefónico de entrada, un conjunto de agentes (extensiones) activas y una extensión de emergencia. Cuando un cliente llama al número de entrada, la centralita desvía su llamada a uno de los agentes activos. El agente elegido es normalmente el que más tiempo lleva sin contestar llamadas. Cuando la conversación finaliza el agente pasa a estar en estado de *work ready*. Eso significa que está haciendo algo para el cliente que acaba de hablar con él (enviar un fax, llamar a otro lugar). Cuando

finalice esa tarea, el agente debe indicarlo con una tecla de función de su teléfono (se deben usar teléfonos digitales de altas prestaciones) para que el sistema sepa que le puede volver a pasar llamadas. Cuando un agente llega al sistema debe darse de alta en un grupo ACD (o en varios grupos) usando también las teclas de función de su teléfono y cuando se va el agente se da de baja en el o los ACD's. La extensión de emergencia es la que suena cuando no hay ningún agente dentro del ACD.

Los estándares CTI (como, como por ejemplo TSAPI) permiten gestionar el ACD. Por ejemplo, se pueden dar de alta y de baja extensiones en los grupos ACD. Se puede saber cuando entran llamadas a un ACD y a qué agente se desvían…

1.3 Dispositivos telefónicos

Hoy en día existen una amplia gama de dispositivos telefónicos que, unidos a un sistema informático, incrementan el valor añadido de los mismos:

- Proporcionando mayor interacción del cliente/usuario con el sistema
- Incrementando la productividad personal
- Añadiendo una nueva dimensión de servicios telefónicos

Entre ellos destacan:

- **Centralitas digitales o PBX**

Las PBX suelen ser el dispositivo principal del 90% de los sistemas CTI. Es importante saber que una centralita sólo va a funcionar según los estándares CTI si el fabricante ha incluido en ella el *firmware* adecuado. Muchas veces para explotar la funcionalidad CTI es necesario adquirir un módulo adicional e instalarlo.

- **Tarjetas propietarias de telefonía**

Son dispositivos pensados exclusivamente para CTI. Se conectan al ordenador por puerto serie o directamente al bus ISA o PCI. También se conectan a una (o varias líneas telefónicas) para realizar alguna función específica.

- **Teléfonos con conexión a PC incorporada**

Algunos fabricantes (como, por ejemplo Bosch) proporcionan estos modelos que siempre hay que conectar a la línea telefónica y, opcionalmente, al ordenador. Desde el PC se pueden realizar diferentes acciones como marcar, contestar o transferir llamadas.

- **Módems**

Es el dispositivo CTI más sencillo pero con una adecuada programación se puede lograr funciones similares a algunas tarjetas propietarias de precio muy superior. Hoy día un buen módem debe tener capacidades de voz (estos módems llevan una tarjeta de sonido integrada, son capaces de emitir audio por la línea y de grabar lo que se recibe por ella), reconocimiento de tonos DTMF que reciba de la línea y enviar/recibir faxes usando los estándares del grupo III y IV.

Históricamente, estos dispositivos tenían un inconveniente común: suponen atarse a la tecnología del fabricante que proporciona un SDK (*Software Development toolKit*) propietario para su programación.

Por tanto, el objetivo a lo largo de la historia ha sido, más por parte de los desarrolladores de software CTI que por parte de los fabricantes, el estandarizar la tecnología CTI, entendiendo por ella todo el conjunto software y hardware que permiten intercomunicar dispositivos telefónicos y sistemas informáticos con el objetivo de proporcionar servicios de valor añadido. De ésta definición se deducen dos principales aspectos necesarios a estandarizar:

1. Un protocolo o protocolos que definan la comunicación entre el dispositivo telefónico y el ordenador. Estos protocolos deben ser suficientemente generales para describir todos los posibles dispositivos telefónicos existentes.

2. Un conjunto de funciones o API que permita a los desarrolladores de software dotar a las aplicaciones de capacidades telefónicas. Dicha API deber ser un *middleware* entre el programa de usuario y el controlador que se comunica con el dispositivo telefónico.

1.4 Arquitecturas Hardware para los sistemas CTI

1.4.1 Acceso directo o "first party"

El acceso directo (acceso en primera persona o "first party CTI") consiste en la conexión directa del ordenador de interés a una línea telefónica. Este modelo es el utilizado por el estándar TAPI y un posible esquema de bloques sería el siguiente:

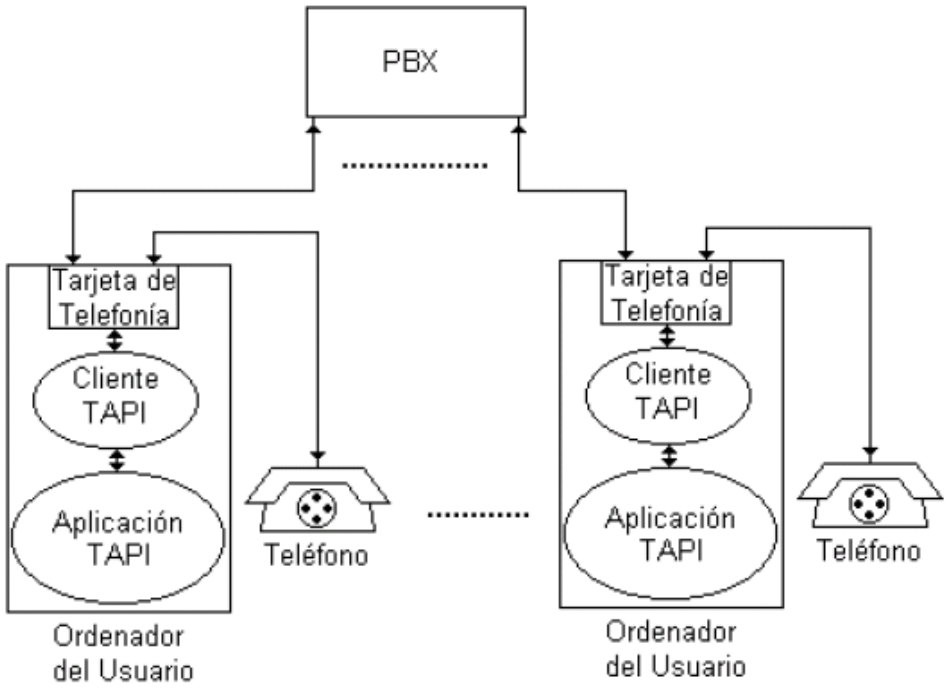


Figura 1.- Arquitectura TAPI *first-party*

En la figura la línea telefónica procede de una centralita privada (PBX) pero puede ser una línea pública procedente de un operador. Para la tarjeta de telefonía hay varias posibilidades que comentaremos en el apartado siguiente. La aplicación se comunica con un soporte TAPI que ya nos proporciona el sistema operativo. Para ser más exactos (aunque volveremos más adelante sobre esto) el soporte TAPI se divide en dos partes: una común proporcionada por el sistema operativo y otra que interactúa directamente con el hardware (esa parte la proporciona el fabricante de la tarjeta y se le llama TSP: *Telephony Service Provider*. El sistema se ha diseñado de tal forma que la aplicación interactúa solamente con la parte común, dejando el TSP oculto. Existe también un estándar que describe las funciones que debe proporcionar un TSP y que debe ser seguido por los desarrolladores de TSP's):

Nótese que la capacidad de la aplicación va a depender totalmente de la señalización presente en la línea que, en este caso es la única fuente de información. Esto es muy importante porque, por ejemplo, muchas PBX no incluyen en la línea las señales que permiten la identificación del llamante (aunque ésta esté presente en las líneas publicas). Por eso, la misma aplicación puede ser capaz de detectar el llamante o no dependiendo de si está conectada a una línea pública o a una centralita.

Otro tipo de limitaciones vienen, a veces, por el conjunto tarjeta-TSP. Nótese que para generar desde la aplicación una llamada es necesario generar las señales correspondientes (tonos DTMF o, incluso, pulsos) e introducirlos en la línea. Todos los dispositivos TAPI son capaces de generar este tipo de tonos pero, por ejemplo, la señal que permite retener una llamada (que es estándar en líneas analógicas) es generada por todos los teléfonos pero por casi ningún dispositivo TAPI. No poder retener implica también no poder transferir (una transferencia se hace reteniendo, marcando y colgando). Hay veces en que las limitaciones vienen del software y no del hardware, incluso hay casos en que cambiando la versión del TSP aumentan las capacidades. Cuando en TAPI intentamos una acción no soportada la llamada a la función correspondiente devuelve siempre estado de

fracaso. Sin embargo, existe un mecanismo que es capaz de averiguar, a priori, cuáles son las acciones permitidas.

Por último, las limitaciones más importantes del acceso "first party" viene de su propia definición. Esto es: cada ordenador que quiera ejecutar aplicaciones CTI necesita una nueva tarjeta y, lo que es mucho peor, cada ordenador conoce solamente las llamadas en las que interviene su línea. Nótese que, en principio, no hay ningún punto en el que se pueda tener información de todas las llamadas que atraviesan la central (como veremos más adelante, hay soluciones a esta limitación).

1.4.2 Diferentes esquemas para el acceso directo

Las primeras soluciones para el acceso "first party" utilizaban un esquema de conexiones muy similar al de la siguiente figura 2:

Figura 2.- Dispositivo TAPI "puro" externo

Este esquema se usaba en productos como, por ejemplo, la TAU-D de Ericsson. La TAU-D permitía que el puesto de operadora de la centralita (que recibía todas, o casi todas las llamadas entrantes) estuviese informatizado. Así se podía grabar un registro de los llamantes, transferir llamadas desde la pantalla del PC… La comunicación de la TAU-D con el ordenador se hacía a través de un puerto serie RS-232. El protocolo lógico de la comunicación era totalmente propietario pero el programador de la aplicación no tenía que conocerlo ya que

un driver TSP proporcionado por Ericsson resolvía ese problema (recordar que el programador se comunica con TAPI y TAPI con el TSP). La TAU-D fue retirada del mercado debido a los problemas que presentaba actualizar el driver con los cambios de versión de Windows (funcionaba en 3.11 y en W95, nunca funcionó en NT). Cuando se retiró (1999), Ericsson ya disponía de soporte TSAPI y estaba empezando a trabajar en el TAPI sobre TSAPI; en ese momento la TAU-D ya era casi "una reliquia del pasado".

Hoy en día el esquema anterior subsiste pero en vez de un elemento propietario y costoso se suele utilizar un módem que es un dispositivo estándar y barato. Al utilizar un módem hay que tener cuidado con el driver TSP que se utiliza. Windows dispone de un driver estándar llamadazo UNIMODEM (fichero Unimdm.tsp, localizado en el directorio SYSTEM) que funciona con cualquier módem estándar que cumple el juego de comandos HAYES. Con el UNIMODEM, los comandos TAPI que funcionan son muy limitados. Los módems avanzados deben traer su propio driver TSP hecho por el fabricante y que soportará funciones avanzadas como generar/reconocer dígitos DTMF…

Nótese que el esquema de la figura 1 corresponde con el uso de un módem interno. Existen otros dispositivos internos que realizan este tipo de funciones. Un ejemplo son las tarjetas fabricadas en Madrid por *Teyma* que además de realizar funciones telefónicas incorporan funciones de reconocimiento y síntesis de voz.

Otro esquema que sigue siendo TAPI "puro" (acceso a una sola línea) es el que permite conectar el teléfono directamente al PC. Este esquema ha sido elegido por el fabricante Bosch para crear aplicaciones CTI de gama baja.

Figura 3. Teléfono conectable a PB

Por último comentaremos que, cada vez más frecuentemente se ha ido rompiendo el esquema TAPI "puro" o básico para pasar a otros más avanzados. Generalmente, se trata de conectar una centralita (con pocas extensiones) a un PC y usar un driver TSP proporcionado por el fabricante de la PBX. En estas instalaciones el software funciona como si se hubieran conectado varios dispositivos telefónicos al ordenador (uno por cada extensión). Esta es la solución adoptada por *Kathrein* o *Elmeg* que fabrican centralitas de gama baja.

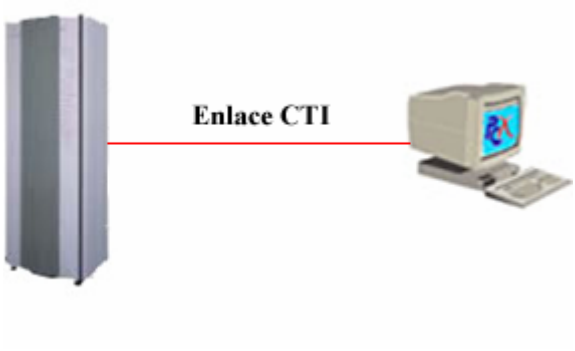


Figura 4.- Centralita TAPI

1.4.3 Acceso indirecto o "third party"

El acceso indirecto se basa en definir un ordenador como servidor de telefonía y conectarlo de alguna forma con la central (ya veremos, más adelante, las formas más usadas). El estándar más importante (no el único) que usa este tipo de acceso es TSAPI. La arquitectura TSAPI, expresada en un esquema de bloques) sería así:

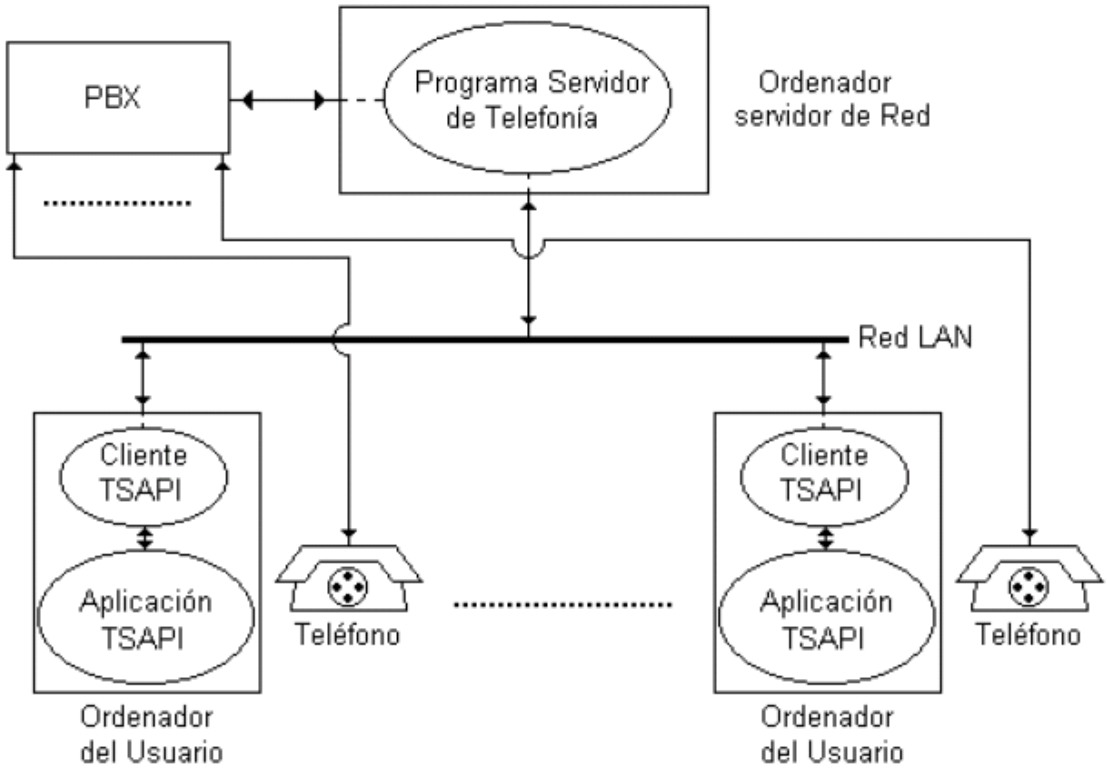


Figura 5.- Arquitectura TSAPI

Como podemos ver en la figura ésta es una arquitectura que demanda más recursos: el TSAPI no es posible concebirlo sin una PBX y además hace falta una infraestructura de red local y un ordenador que actúe como servidor. Sin embargo, estos requerimientos no son en absoluto excesivos para cualquier empresa de tamaño medio, la red y el servidor o servidores ya suelen existir. Podemos comentar que las aplicaciones se comunican con un cliente TSAPI que suele distribuir el fabricante de la central (aunque teóricamente pertenece a los creadores de TSAPI: Novell y ATT). Este cliente utiliza la red para comunicarse con el servidor. En las primeras versiones el servidor era obligatoriamente una máquina corriendo *Novel Netware* y el protocolo de red para las funciones de telefonía era IPX. Actualmente, existen versiones que permiten instalar un servidor con Windows-NT (ó 2000 Server) y en la comunicación se define un servicio de red basado en el protocolo UDP (*User Datagram Proto*col). El puerto UDP a usar es configurable aunque es típico el 2555. El programa servidor es facilitado por el fabricante de la PBX.

No todas las PBX admiten este tipo de arquitectura CTI sino sólo las de gama más alta. TSAPI todavía se considera una solución "de lujo".

Aunque TSAPI define todas las opciones a implementar y, en principio, son obligatorias, en la práctica sucede que puede haber funciones no implementadas (por supuesto, no las más comunes). A pesar de esto, TSAPI es muchísimo más funcional que TAPI. En TAPI es muy raro el sistema capaz de retener, transferir o soportar conferencias mientras que en TSAPI esas opciones se soportan siempre.

Nótese que "por definición" y, al contrario que en TAPI, sí existe información centralizada de todo lo que ocurre en la PBX. Es más, el servidor permite a cualquier ordenador consultar toda esa información. TSAPI controla el acceso "en base a usuarios"; cuando una aplicación se empieza a comunicar con TSAPI necesita proporcionar el login y la clave de un usuario para autenticarse frente al servidor. Si el usuario tiene acceso al servicio de telefonía, tendrá acceso a todo (podrá genera una llamada en cualquier teléfono, sabrá todas las llamadas que hay en la PBX, etc). Esta permisividad excesiva se basa en dejar a las aplicaciones la responsabilidad de controlar qué cosas se deben hacer y cuáles no.

Ericsson estaba desarrollando una aplicación llamada TAPI sobre TSAPI. Se trata de un TSP que reencamine las peticiones de aplicaciones TAPI al cliente de red de TSAPI. De esta forma tendríamos una arquitectura TSAPI (la más potente) pero también podríamos utilizar aplicaciones TAPI (que son las más numerosas hoy día debido al empuje de Microsoft).

1.4.4 Diferentes esquemas para el acceso indirecto

El esquema físico para el acceso indirecto (o particularmente, para el TSAPI) es idéntico al de la figura 4. Sin embargo, las diferencias lógicas son muy grandes. En una centralita gobernada con TAPI cada extensión es (desde el software) un dispositivo independiente, es la "solución" adoptada para gobernar una central con un estándar pensado para dispositivos telefónicos individuales

(monolínea). TSAPI está pensado desde el principio para gobernar una central. En general, TSAPI es un estándar aplicable a centrales de gama más alta que TAPI. El enlace entre la central y el servidor de telefonía puede adoptar diferentes formas. Las últimas tendencias es conectar directamente la PBX a la red IP, es decir, incluir una interfaz de red local en la PBX). Así la central puede correr su propio servidor TSAPI o comunicarse con uno a través de la red. Esta arquitectura da además nuevas posibilidades utilizando tecnologías de voz sobre IP.

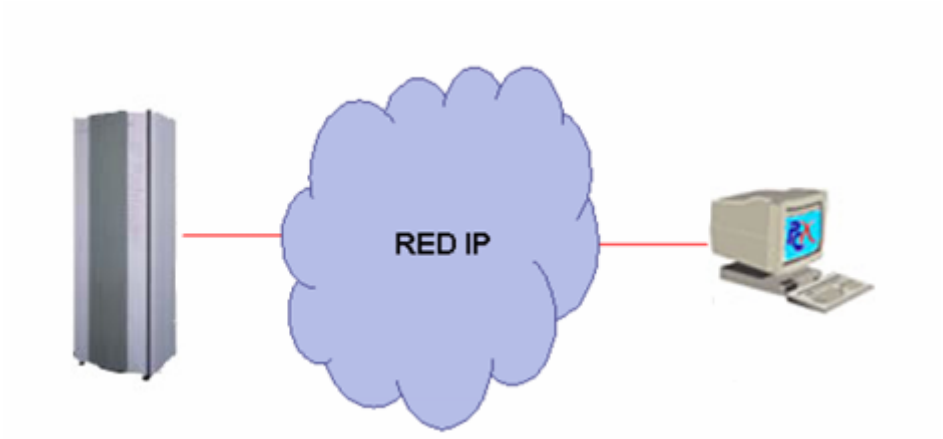


Figura 6. Conexión PBX-PC a través de la red IP

1.5 Arquitecturas Software

1.5.1 Capa única (lectura de tramas de tarificación)

Aunque se haya titulado a este apartado "capa única", evidentemente, el número de capas depende la aplicación que diseñemos. En este apartado nos referimos a las aplicaciones totalmente autónomas, esto es: las que no usan un framework de programación como pueden ser TAPI o TSAPI. En este caso el programador se ve obligado a programar todos los detalles de la comunicación con los dispositivos telefónicos. El ejemplo más habitual suelen ser las aplicaciones de tarificación. Para este tipo de aplicaciones no existe un estándar aceptado que ayude al programador a salvar las diferencias entre fabricantes por

lo que, normalmente, será necesaria la lectura directa de las tramas de tarificación entregadas por la central.

En estos casos, el programador debe intentar que su programa sea adaptable al mayor número de equipos posible. Por ejemplo, son comunes las centrales en las que el protocolo de tarificación funciona sólo en el sentido de la PBX al ordenador, se envía una trama por llamada y el ordenador nunca emite datos (existen protocolos más complejos en los que el ordenador tiene que solicitar la información y/o confirmar su correcta recepción). En este caso sencillo, lo único que cambia entre fabricantes es el formato de la trama y es fácil hacer un programa que se adapte a diferentes formatos (evidentemente, el programa debe conocer el formato de cada fabricante y el usuario debe indiciar el modelo de PBX que está utilizando).

Aunque es posible tarificar utilizando TAPI o TSAPI, no se consigue la fiabilidad obtenida con las tramas específicas para tarificación, esto se debe a que las duraciones de las llamadas o el número destino no siempre se obtienen de forma fiable con TAPI o TSAPI (sobre todo en casos de líneas de salida analógicas o GSM). Además, no tendríamos la ventaja del buffer de tarificación que incorporan muchas centrales; esta memoria guarda las tramas no procesadas para enviarlas después (por ejemplo, mientras reiniciamos el ordenador tarificador). Un buffer de este tipo puede llegar a guardar (según el tráfico de la central) días enteros de actividad.

1.5.2 El modelo cliente/servidor en CTI

Al utilizar estándares muy elaborados como TAPI, TSAPI u otros, podemos decir que nuestra aplicación es cliente de unos servicios de telefonía previamente instalados en la máquina. En este caso, el programador utiliza un *framework* de programación consistente en una API's más un conjunto de librerías que le hacen transparente la comunicación con el dispositivo telefónico. Además, ahora la misma aplicación es (casi) independiente del dispositivo telefónico ya que cada fabricante implementa en el driver TSP las diferencias en la comunicación entre

el dispositivo telefónico y el PC manteniendo las mismas interfaces, ocultando así la comunicación al nivel de aplicación.

Capítulo 2
El estándar TAPI y el protocolo CSTA

2.1 Introducción

TAPI, *Telephony Application Programming Interface* [1], es un conjunto de librerías de programación que permiten interactuar a un computador y a un dispositivo telefónico. TAPI es bastante compleja; soporta transmisión de voz y datos e incluso tiene capacidades de control de llamadas como llamada en espera o mensajes de voz. TAPI actúa como una capa de abstracción entre el nivel de aplicación y la comunicación con el dispositivo telefónico pero por sí sola no proporciona ninguna funcionalidad útil. Como capa de abstracción, su función es proporcionar un modelo de programación independiente del dispositivo y consistente. En Win32, una aplicación no necesita conocer si está dibujando una línea con una tarjeta VGA 16 colores o una tarjeta aceleradora 3D super-VGA con 16 millones de colores. Todo lo que necesita es conocer es donde quiere dibujar la línea. TAPI pretende hacer para la telefonía lo que GDI hace en el aspecto gráfico.

Microsoft diseñó TAPI para ofrecer una alternativa a soluciones CTI dependientes del hardware. Con TAPI, las aplicaciones no necesitan saber nada sobre el hardware subyaciente.

2.2 Arquitectura TAPI: Windows Open Services Architecture (WOSA).

TAPI está basado en los principios de la Arquitectura de servicios abiertos de Windows (*Windows Open Services Architecture*). El concepto de WOSA es el de proporcionar un interfaz uniforme de programación (un API estándar) que actúe como capa de abstracción entre el nivel de aplicación y la implementación de los servicios requeridos.

El modelo WOSA está formado por tres piezas fundamentales. Cada una de estas piezas juega un importante, e independiente, rol en proporcionar servicios a las aplicaciones. Estos componentes son:

- El API de cliente: el conjunto de llamadas a funciones usado por la aplicación que requiere el servicio.
- El proveedor de servicios (*Service Provider):* Son módulos o drivers que implementan un servicio en concreto. Todos ellos deben ser implementados según un *SPI (Service Provider Interface)*, es decir, deben implementar un interfaz definido de funciones que será usado indirectamente por las aplicaciones.
- Interfaz API/SPI: Módulo que enlaza las llamadas al API de cliente desde la aplicación con las llamadas al SPI. Generalmente está implementado como una biblioteca de enlace dinámico (DLL) en el entorno Windows.

La arquitectura TAPI según la WOSA se resume en la figura 7.

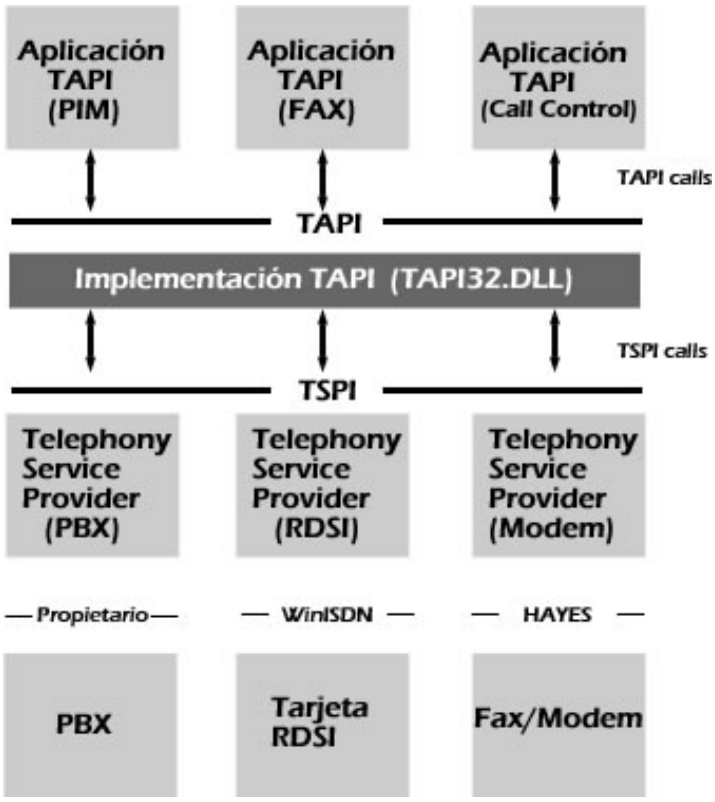


Figura 7. Arquitectura TAPI / WOSA

En ella podemos identificar los tres componentes descritos anteriormente:

- API de cliente: El conjunto de llamadas que proporciona acceso a servicios de telefonía es la *Telephony API (TAPI).*
- Proveedor de servicios: Al proveedor de servicios en TAPI se le denomina comúnmente TSP (*Telephony Service Provider).* Normalmente el proveedor de servicios de telefonía es implementado como una biblioteca de enlace dinámico aunque es común que ésta tenga extensión TSP. Todos los TSP deben implementar el *TSPI (Telephony Service Provider Interface)* que define el interfaz de funciones que deben proveer todos los driver de telefonía. El TSP es el módulo (controlador) encargado de implementar la comunicación con el dispositivo telefónico y normalmente es proporcionado por el fabricante del mismo.

- Interfaz API/SPI: Implementado en la biblioteca de enlace dinámico TAPI32.DLL. Se encarga de mapear las llamadas a TAPI desde la aplicación a llamadas al driver TSP que se traducirán en una acción sobre el dispositivo telefónico.

Con este modelo, como programadores de aplicaciones sólo debemos preocuparnos por conocer el API de cliente, es decir, qué funciones están disponibles y cómo usarlas. Paralelamente, como programadores de TSP's, debemos conocer la especificación *TSPI* para saber las funciones que debe implementar nuestro TSP y su interfaz. El TSP, una vez instalado y registrado en el sistema como proveedor de servicios de telefonía , está disponible para su uso desde la aplicación TAPI.

2.3 TAPI 1.x

Windows 95 viene con TAPI versión 1.x. Como muchos otros componentes de Windows 95, su implementación interna es de 16-bit, por lo que las aplicaciones de 32-bit enlazan con la biblioteca de enlace dinámico TAPI32.DLL que actúa como capa intermedia entre la aplicación y TAPI.DLL, de 16-bit. La siguiente figura ilustra la arquitectura interna de TAPI 1.x:

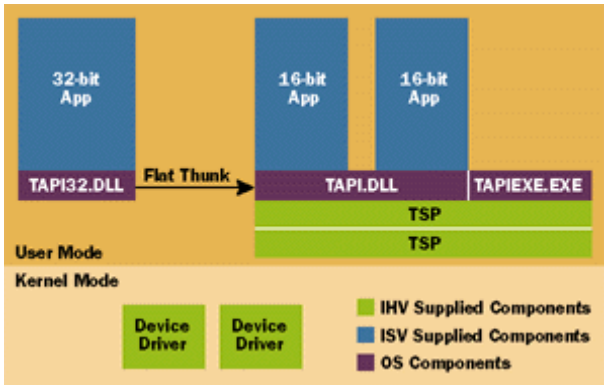


Figura 8.- Arquitectura TAPI 1.x

Tapi.dll es cargada en una VM (virtual machine) de 16-bit junto con un proceso llamado Tapiexe.exe. Tapi.dll hace de servidor para las aplicaciones pero, a su vez es cliente del servidor TAPI principal (TAPIEXE) y éste es cliente del driver TSP (**fichero.tsp**, *Telephony Service Provider*) que es quien de verdad ejecuta las acciones sobre los dispositivos.

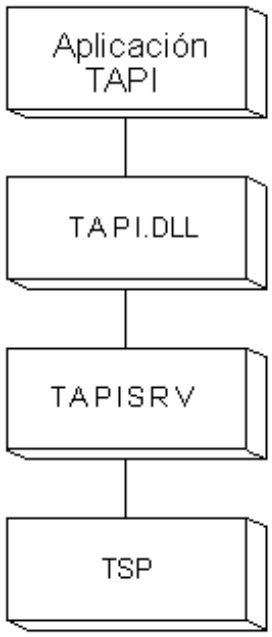


Figura 9.- Capas en TAPI 1.x

La razón de que se introduzca el módulo TAPIEXE (y en versiones posteriores, TAPISRV) entre la DLL y el TSP es que en un mismo sistema pueden convivir varias versiones de la DLL TAPI (normalmente se trata de **tapi.dll** y **tapi32.dll**). Así, se pueden tener aplicaciones TAPI de 16 y 32 bits funcionando a la vez.

2.4 TAPI 2.0

Windows NT 4.0 introdujo TAPI 2.0, implementada completamente en 32 bits, que ofrece mejoras significativas de estabilidad y confiabilidad. En la figura 10 se ilustra la arquitectura de TAPI 2.0. Para los programas de aplicación, los

cambios son transparentes. TAPI 2.0 soporta aplicaciones TAPI 1.3 y TAPI 1.4 al igual que aquellas escritas específicamente para TAPI 2.0.

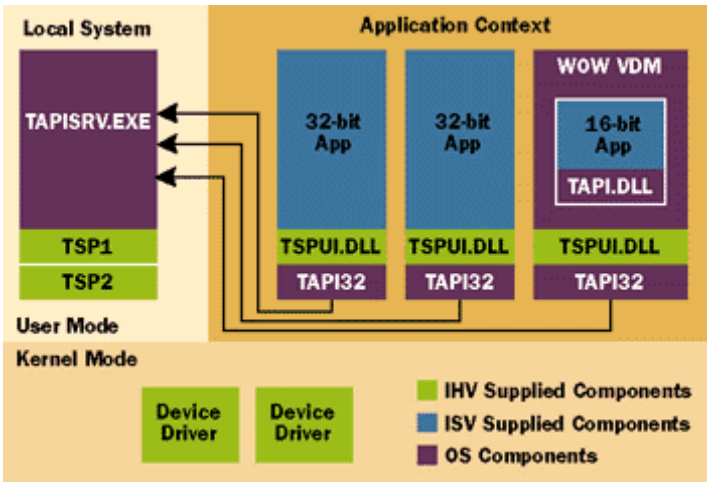


Figura 10.- Arquitectura TAPI 2.0

Por otra parte, todos los TSPs deben ser bibliotecas de enlace dinámico de 32 bits usando Unicode. En TAPI 2.0 el proceso de ayuda TAPIEXE.EXE es sustituido por TAPISRV.EXE, que corre como servicio dentro del contexto del sistema local. Debido a esta arquitectura, un TSP no puede llevar a cabo funciones como mostrar un cuadro de diálogo en la pantalla avisando de cualquier contratiempo. Por ello, la versión 2.0 introdujo una DLL secundaria de interfaz de usuario. Esta UI DLL contiene elementos de interfaz gráfico específicos del TSP y es cargada en contexto de aplicación. A través de esta biblioteca, el TSP puede mostrar cuadros de diálogo y otros elementos de interfaz de usuario ya que el contexto de aplicación sí tiene acceso al escritorio.

2.5 TAPI 2.1

Tapi 2.1 es una actualización *post-release* que añade el modelo cliente-servidor a Tapi 2.0. Como comentamos en el capítulo anterior, la limitación más

importante del acceso "first party" de TAPI es que cada ordenador "ve" solamente las líneas telefónicas a las que se encuentra conectado físicamente. Consideremos una empresa con una PBX con 4 extensiones, cada una asociada a un empleado que dispone en el escritorio de un PC.

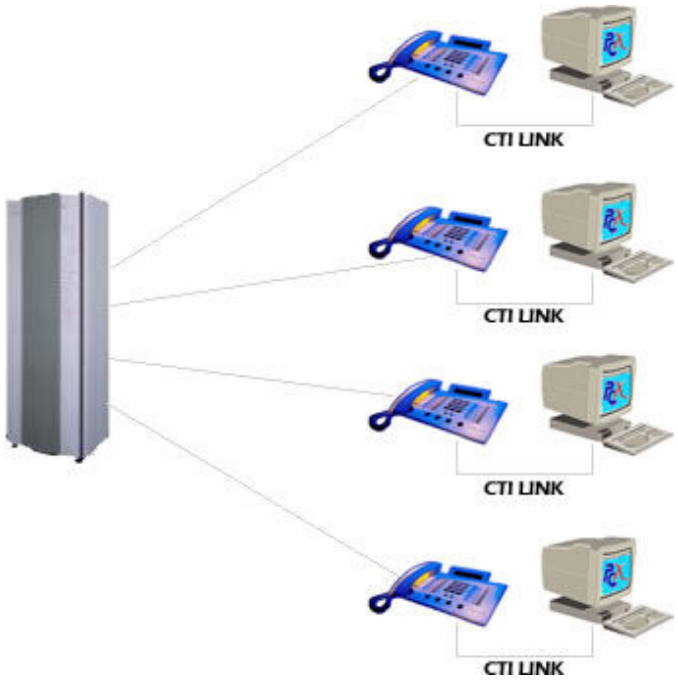


Figura 11.- First.-party puro

Con TAPI 2.0, es decir, con una arquitectura "first-party" pura, cada PC puede tener instalado el TSP que controla el teléfono digital y establecer un enlace CTI con él. De este modo, será capaz de reproducir en pantalla toda la funcionalidad del teléfono (toda la que implemente el TSP) y llevar a cabo acciones sobre esa extensión, tales como marcar, colgar, transferir, etc. En ningún caso es capaz de controlar las llamadas del resto de extensiones de la PBX.

Otra opción (dentro todavía de "first-tapy" puro) sería crear un enlace CTI entre la PBX y un PC que estuviera corriendo el TSP de la centralita:

Figura 12.- Otro posible enlace first-party puro

Con esta arquitectura, el PC sería capaz de "ver" todas las extensiones de la PBX. Este modelo es, por tanto, óptimo para aplicaciones de tarificación, registro o gestión de llamadas. Sin embargo, es imposible conseguir un modelo que permita ejecutar aplicaciones CTI de escritorio tal y como muestra la figura 12.

TAPI 2.1 permite a las aplicaciones TAPI en PCs cliente acceder transparentemente a un TSP corriendo en un servidor remoto tal y como si ese TSP estuviera instalado localmente en el sistema. Esta es la única diferencia entre TAPI 2.0 y TAPI 2.1 ya que la versión 2.1 no introduce nuevas funciones del API ni requiere un nuevo SDK (*Software Development ToolKit*). Para conseguir esta nueva funcionalidad sin modificar el API, Microsoft introdujo un TSP "ficticio" llamado **remoteSP.tsp** quien introduce el funcionamiento remoto en un sistema pensado para funcionamiento local. El TSP **remoteSP.tsp** se encarga de realizar la comunicación con el servicio de telefonía del servidor vía RPC y solicitarle las peticiones hechas por el cliente así como devolverle el resultado procedente del servidor. Esta comunicación es totalmente transparente para el cliente quien cree tener un enlace CTI directo con el dispositivo telefónico.

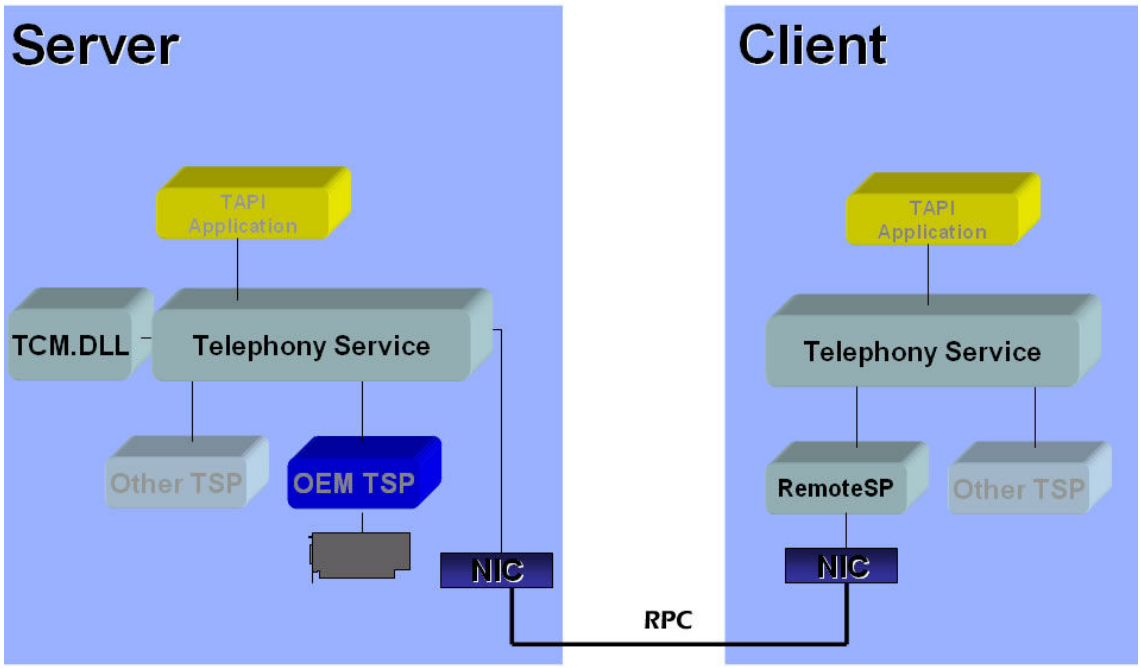


Figura 13.- Tapi 2.1

Para utilizar este esquema es necesario configurar en el cliente a qué dispositivos queremos acceder en la máquina remota (con la aplicación **tcmsetup.exe). tcmsetup.exe** también se usa para configurar el servidor como tal. Para el control de accesos, esto es: indicar a qué usuarios se les permite acceder y a qué dispositivos, se usa **tcmapp.exe.** En la figura podemos ver un elemento que no habíamos comentado hasta ahora: las **Telephony Client Management Dll's (TCM.DLL),** se trata de programas que filtran todas las llamadas al sistema telefónico efectuadas desde ordenadores remotos. Estas dll's pueden autorizar o denegar cada acción. Por defecto no hay ninguna instalada con lo que todas las acciones se permiten. Normalmente serán desarrolladas por el mismo programador que las aplicaciones CTI en uso.

Por último destacar que la principal limitación de TAPI Remoto es que permite control de llamadas pero no de contenidos, es decir, no permite acceder al flujo de información de llamadas. Esto impide a aplicaciones como detección/generación de tonos DTMF utilizar este esquema.

Teniendo en cuenta TAPI 2.1 nuestro ejemplo anterior podría plantearse según la siguiente arquitectura:

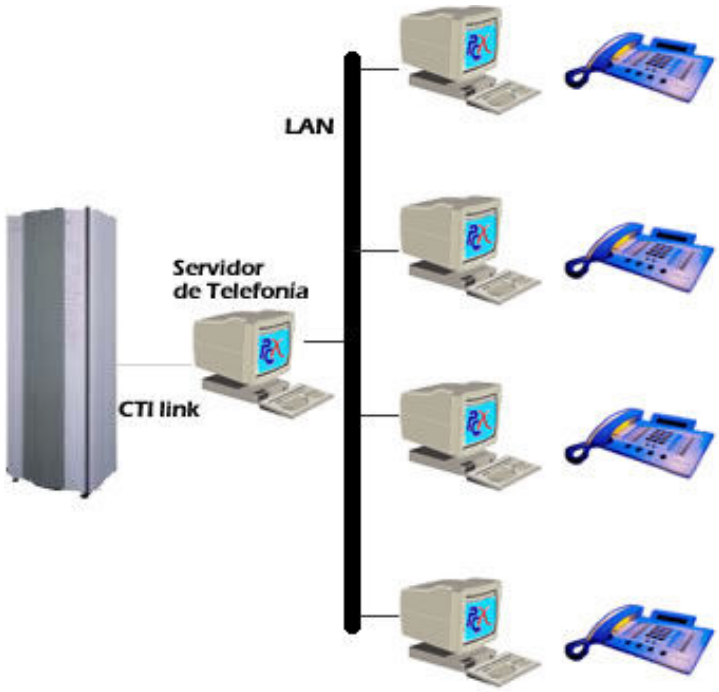


Figura 14.- Arquitectura Remote Tapi

Con este modelo y definiendo bien los permisos en el servidor de telefonía, cada PC cliente "verá" tantas extensiones como se le hallan asignado en el servidor.

2.6 TAPI 3.x

TAPI 3.x es una nueva API que incorpora capacidades de telefonía IP a las versiones anteriores de TAPI. La API está implementada como un conjunto de objetos COM (*Component Object Model*). Gracias a este modelo, se pueden desarrollar aplicaciones TAPI en muchos lenguajes como Java, Visual Basic o C/C++. Además de estas novedades, la arquitectura también se ve modificada:

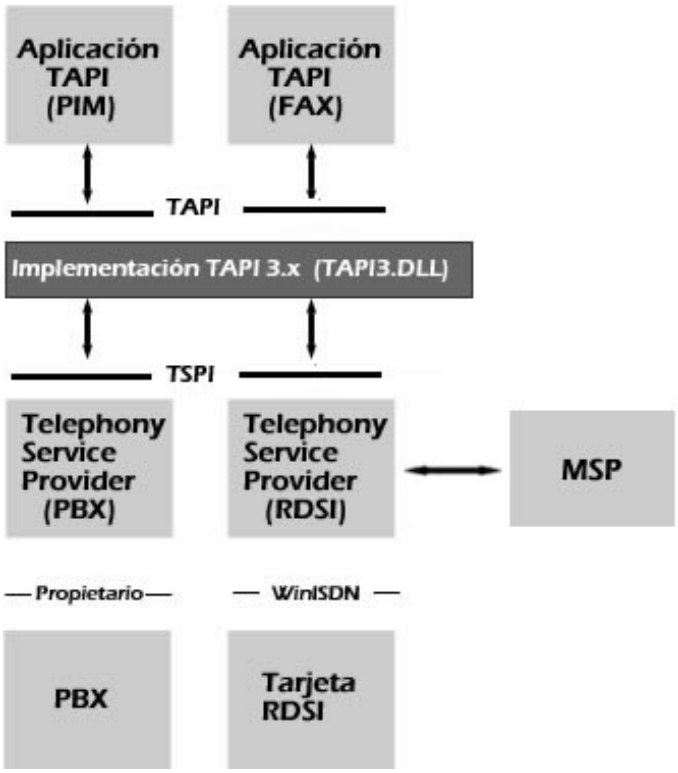


Figura 15.- Arquitectura TAPI 3.x

Como vemos, se introduce un nuevo bloque: MSP. MSPs o *Media Service Providers*) proporcionan una forma uniforme de acceder al flujo de información en una llamada soportando el API *DirectShow*. Dicho en otras palabras, un MSP implementa los interfaces DirectShow para un TSP en particular.

2.7 Estándar de conexión telefonía ordenador. Protocolo CSTA

CSTA [5] es un acrónimo de *Computer Supported Telecommunication Applications* y fue desarrollado por la ECMA (*European Computer Manufacturers Association*). CSTA se suele asociar a TSAPI (de hecho, la dll cliente de TSAPI se suele llamar **csta32.dll**), sin embargo, dicha creencia es un error. La verdad es que CSTA describe un protocolo de comunicación entre un ordenador y un dispositivo telefónico. Dicha descripción se divide en dos documentos:

- La descripción de los servicios que el sistema telefónico ofrece al ordenador (posibles acciones a realizar) y también de los eventos que se enviarán cuando ocurran determinados sucesos en el sistema telefónico.

- La descripción detallada de las tramas de datos intercambiadas (*Application Protocol Data Units*, APDUs) para cada servicio y para cada evento. Este tipo de descripciones se pueden hacer en ASN.1 y también en XML.

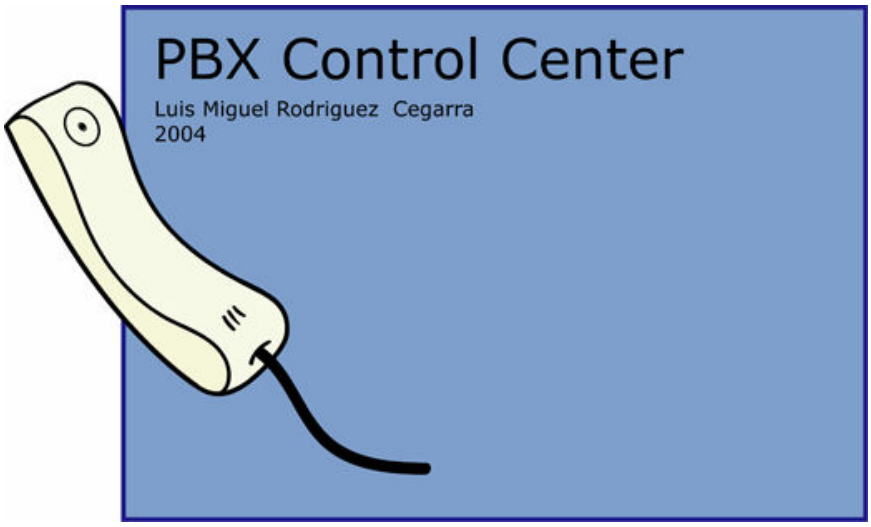
El protocolo CSTA se encuentra en la versión 3 (llamada por ECMA fase III), los documentos que lo definen son ECMA-269 (servicios) y ECMA-285 (protocolo). Para la primera versión (fase I) los documentos eran ECMA-179 (servicios) y ECMA-180 (protocolo). Todos ellos se pueden descargar del sitio Web de la ECMA: http://www.ecma.ch (más concretamente, la dirección de la página dedicada al CSTA es:

http://www.ecma.ch/ecma1/TOPICS/TC32/TG11/CSTA.HTM).

CSTA es, por tanto, un conjunto de acciones y eventos que se pueden invocar/detectar desde un ordenador conectado a un dispositivo telefónico. Es un estándar libre que cualquier fabricante de dispositivos puede adoptar en sus productos. CSTA podría ser utilizable desde un programa que implementase la correspondiente comunicación siguiendo los estándares anteriores. Sin embargo, lo habitual es que los programadores utilicen programas preexistentes que ocultan los detalles de comunicación, empaquetamiento de APDU's... estos programas ofrecen una API (*Application Programs Interface*) consistente en una colección de funciones que es mucho más fácil de utilizar. Este tipo de software intermedio se suele llamar "middleware" y TSAPI es un ejemplo de middleware. TSAPI es una traducción casi directa de los servicios y eventos CSTA y por eso se les suele identificar pero es posible que el middleware sea un dirver TSP y estemos usando un dispositivo CSTA a través de TAPI. De hecho, según documenta la ECMA, el CSTA se ha usado (hasta ahora) con los siguientes

middlewares: TAPI (Microsoft), TSAPI (Novell, ATT y otros), JTAPI (Sun y otros), JavaPhone (Sun) y CallPath (IBM).

Capítulo 3
Manual de usuario de PBX Control Center



3.1 Bienvenido a PBX Control Center

PBX Control Center es un administrador para centralitas digitales (PBX). Ofrece funciones de monitorización en tiempo real del tráfico cursado por la PBX así como funciones de control y administración de las llamadas.

PBX Control Center ofrece además herramientas de auditoría off-line del tráfico cursado por la centralita, tales como Informes de tráfico, Informes de costes o generación de gráficos estadísticos.

Con PBX Control Center podrá mantener una amplia base de datos de todo el tráfico generado en la centralita para posteriormente analizarla mediante un sencillo y amigable interfaz de usuario. Del mismo modo, todos los informes

generados son archivados para su posterior revisión desde el propio interfaz, gracias al visor de informes integrado.

3.2 Requisítos mínimos

Hardware

- **PBX**

PBX Control Center funciona con cualquier centralita digital que implemente la especificación de servidor TAPI (*Telephony API*) de Microsoft. TAPI es una API general que cubre una amplia gama de dispositivos telefónicos proporcionando un modelo de programación que abstrae la comunicación entre estos dispositivos y la aplicación. PBX Control Center está diseñado e implementado siguiendo este modelo y utilizando esta API. El dispositivo telefónico ó PBX debe implementar como mínimo la versión 2.0 de TAPI. El fabricante de la PBX debe proporcionar un driver TAPI (TSP, *Telephony Service Provider*) que permita la interoperabilidad con la aplicación.

No es necesaria ninguna configuración específica de la centralita para poder usar PBX Control Center.

- **Enlace PBX - PC**

La conexión entre la centralita y el PC (enlace CTI) se puede llevar a cabo de diversas formas dependiendo de la centralita y las capacidades del driver (TSP) proporcionado por el fabricante. Remítase a la documentación del fabricante.

- **PC**

- Memoria RAM recomendada: 128 MB

Software

- **Sistema Operativo**

Microsoft Windows NT 4.0, Windows 2000, Windows XP o superior.

3.3 Ventana principal

La ventana principal está dividida en 4 secciones:

- Barra de herramientas
- Menú de vistas
- Vista principal
- Cuadro de estado del sistema

Barra de herramientas

Botón 'Cuadro de diálogo líneas bloqueadas' 

Muestra o esconde la ventana de diálogo que lista las extensiones bloqueadas.

Botón 'Monitor en tiempo real' 

Estando en la vista de monitorización, muestra o esconde el monitor en tiempo real del tráfico cursado por la centralita.

Botón ' Generación de gráficos e informes estadísticos' 

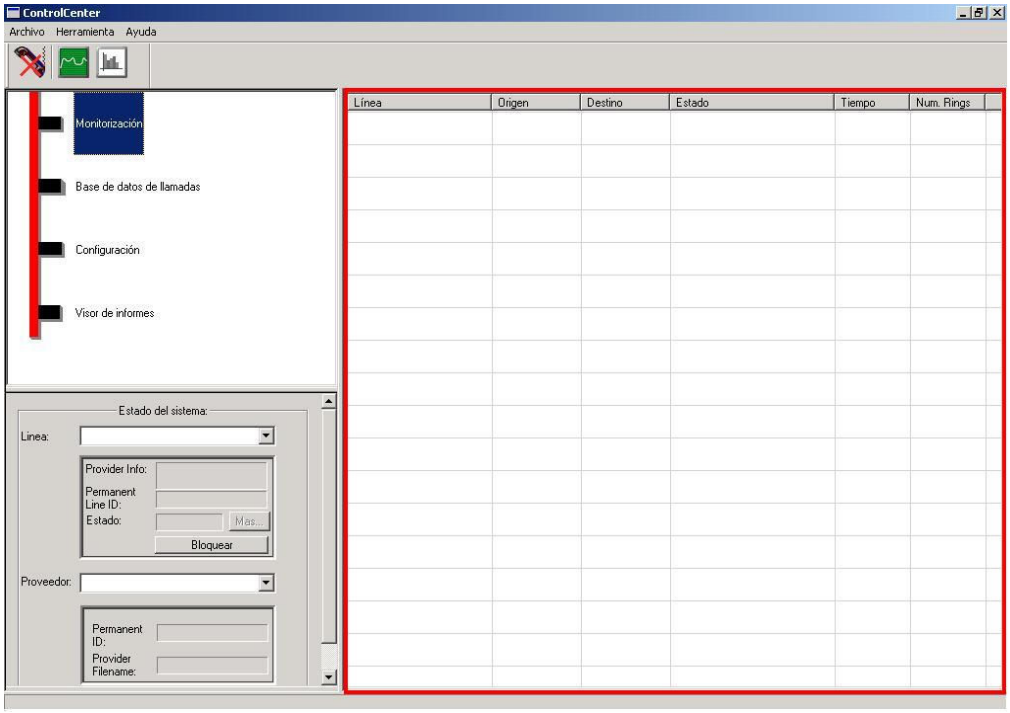
Estando en la vista de Base de datos de llamadas, abre la ventana de diálogo que permite generar varios tipos de gráficos estadísticos e informes del tráfico.

Menu de vistas

Permite cambiar la vista principal de la ventana principal entre los diferentes módulos de que dispone la aplicación:

Monitorización, Base de datos de llamadas, Configuración y Visor de Informes.

Vista Principal



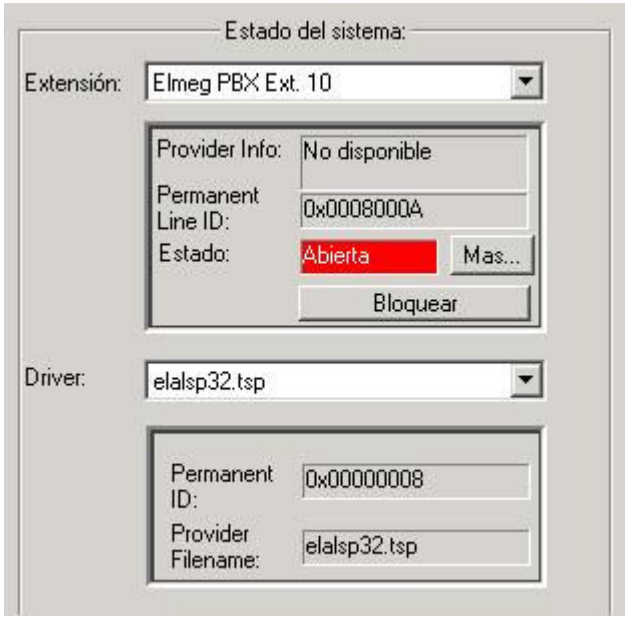
La vista principal de la aplicación cambia según la opción del menú de vistas seleccionado.

Cuadro de estado del sistema

En él podemos ver información sobre cualquier extensión de la PBX relacionada con las características del dispositivo telefónico que la gobierna. Pulsando el botón Mas... de cada extensión se abrirá un cuadro de diálogo con estas características así como otro tipo de información a nivel de programación.

El campo "Estado" muestra en todo momento si la extensión está:

- Abierta la aplicación ha negociado exitósamente con el driver tener el control sobre ella y la extensión no ha sido bloqueada.
- Bloqueada la extensión ha sido bloqueada desde la aplciación. Cuando una extensión está bloqueada no puede realizar ni recibir llamadas de ningún tipo.



Driver: El driver es el controlador de la centralita. Es proporcionado por el fabricante y es necesario para que PBX Control Center se comunique con ella. Gracias a esta arquitectura, PBX Control Center es capaz de funcionar con cualuquier centralita compatible con TAPI (mínimo 2.0) y que proporcione un

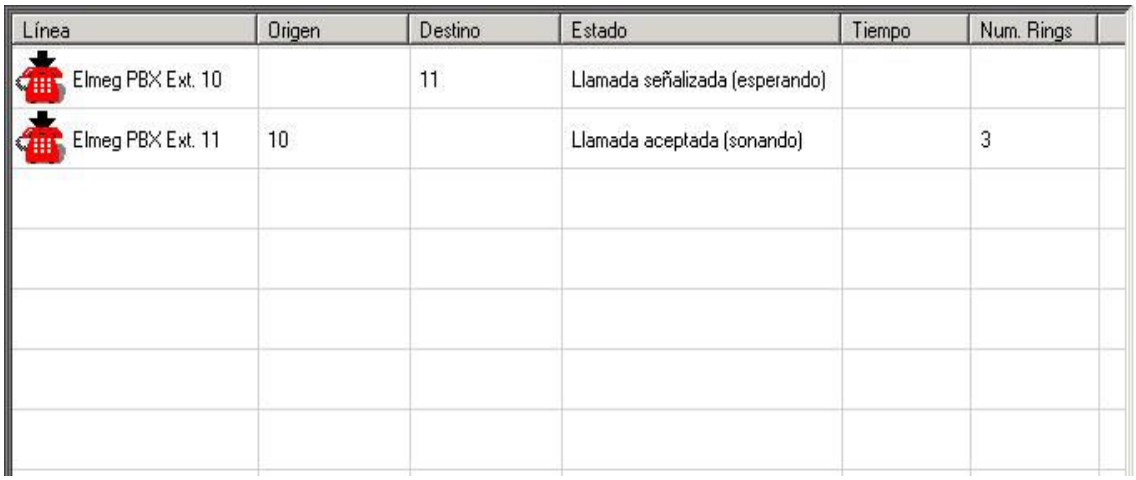
driver (TSP).

El driver es el encargado de hacer visibles al sistema operativo (y en consecuencia, a PBX Control Center) las extensiones, líneas externas y/u otras características de la centralita.

En el sistema operativo, pueden haber tantos drives TAPI como se desee. Es decir, la aplicación puede controlar al mismo tiempo extensiones de distintas centralitas digitales. Además , en el sistema operativo Microsoft Windows vienen instalados varios drivers TAPI estándar para controlar dispositivos telefónicos como modems análogicos, dispositivos de VoIP, etc, por lo que PBX Control Center es capaz también de monitorizar y administrar estos dispositivos.

NOTA: Cuando seleccionamos en la lista desplegable una extensión, automáticamente se selecciona en el desplegable inferior el driver de la centralita que la gobierna.

3.4 Vista 'Monitorización'



En esta vista se monitorizan todos los eventos telefónicos que envía la centralita. Un evento telefónico es cualquier cambio de estado en los dispositivos telefónicos. En el caso de una centralita, será cualquier cambio de estado en las extensiones, líneas, etc. Algunos ejemplos de eventos son:

- Tono de marcado detectado
- Detección de marcado
- Señalización de llamada realizada
- Detección de llamada entrante
- Fin de llamada
- ....

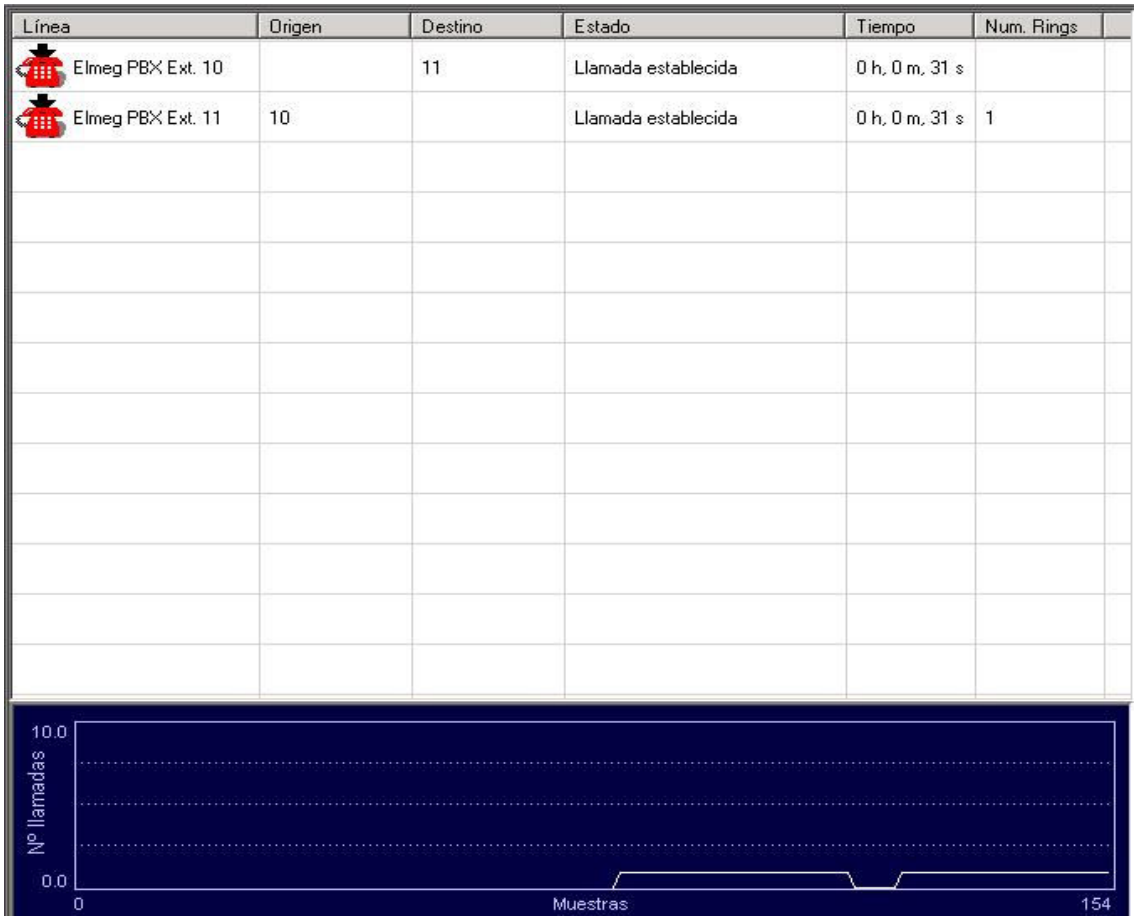
El significado de cada columna en la tabla es el siguiente:

Columna	Descripción
Línea	Indica la línea causante del evento
Origen	En el caso de eventos de llamada entrante muestra el número llamante

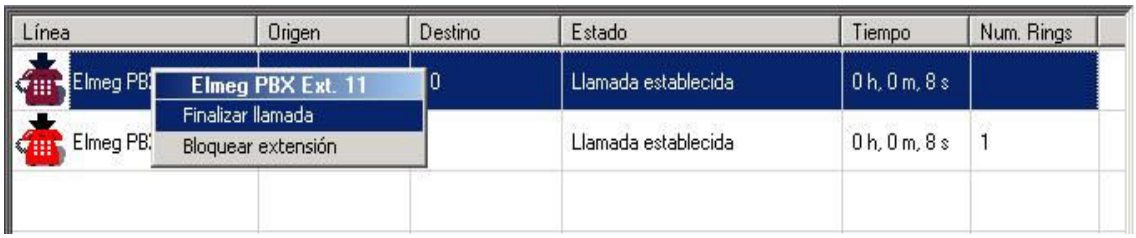
Destino	En el caso de eventos de marcado o llamada realizada, muestra el número destino
Estado	Es el tipo de evento
Tiempo	Contador de tiempo
Num. Rings	Contador de número de "rings" en el momento de la señalización de una llamada

Gráfico de llamadas en tiempo real

Estando en esta vista se habilita en la barra de herramientas el botón  Pulsando en él ampliamos la vista introduciendo un gráfico en tiempo real de las llamadas establecidas en la centralita.



Opciones de control de las llamadas



En cualquier momento haciendo click con el botón secundario del ratón sobre cualquier entrada de la lista de monitorización se desplegará un menú popup con 2 opciones:

- **Finalizar llamada:** Independientemente de si la entrada de la lista corresponde a una llamada establecida o no, la extensión asociada recibe tono de fin de llamada forzando a colgar. Por tanto, si existía una llamada en curso, ésta será finalizada.

- **Bloquear extensión:** Esta opción tiene el mismo efecto que pulsando el botón "Bloquear/Desbloquear" del Cuadro de Estado del Sistema de la ventana principal. Una extensión bloqueada es incapaz de recibir o realizar llamadas. Incluso es incapaz de marcar un número (señalizar una llamada). Si la extensión está bloqueada en el momento de desplegarse el menu popup, esta opción estará marcada con un tick:

Entonces, seleccionar de nuevo esta opción supone desbloquear la extensión.

NOTA: Bloquear una extensión mientras ésta mantiene una llamada en curso no supone finalizar dicha comunicación. El bloqueo se aplica a las posteriores llamadas.

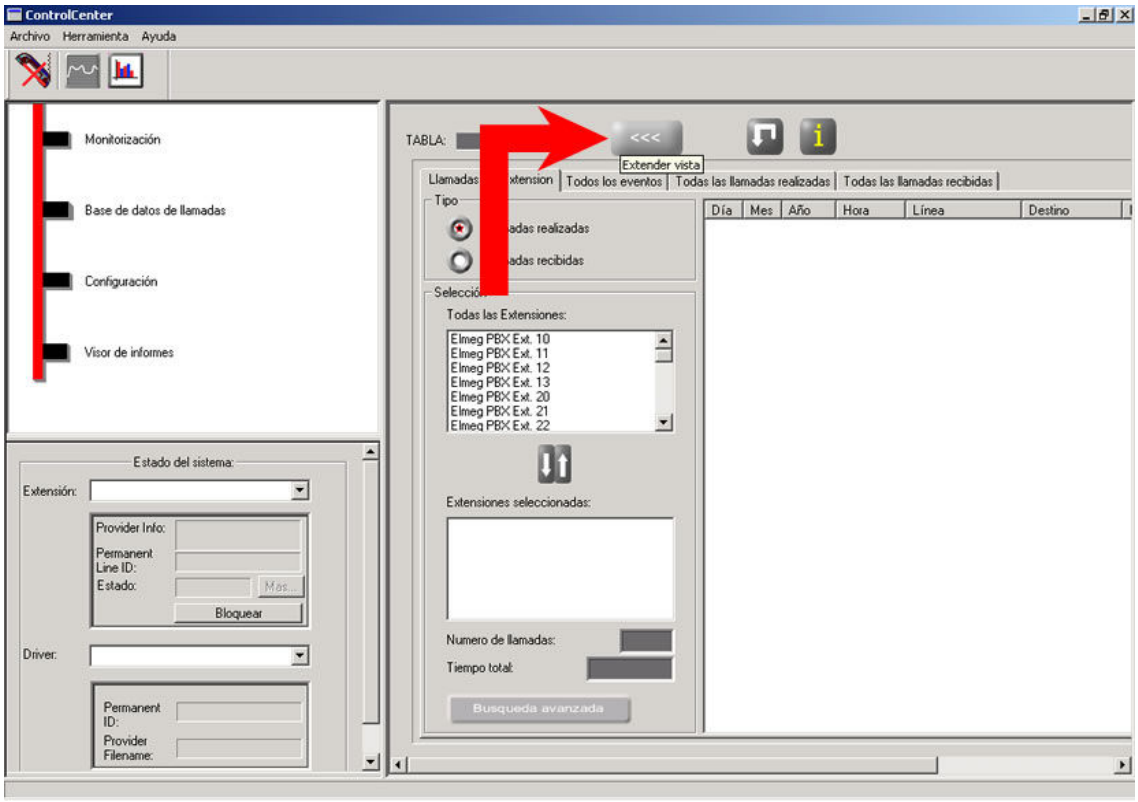
Hasta ahora, hemos visto que existen 2 formas de bloquear una extensión: mediante el menu desplegable de la lista de monitorización o a través del Cuadro de Estado del Sistema. Además, en todo momento podemos ver la lista de extensiones bloqueadas pulsando el icono  de la barra de herramientas. Aparecerá entonces un cuadro de dialogo en la esquina inferior derecha de la ventana principal mostrando las extensiones bloqueadas.

Podemos desbloquear cualquier extensión de la lista seleccionándola y pulsando el botón desbloquear.

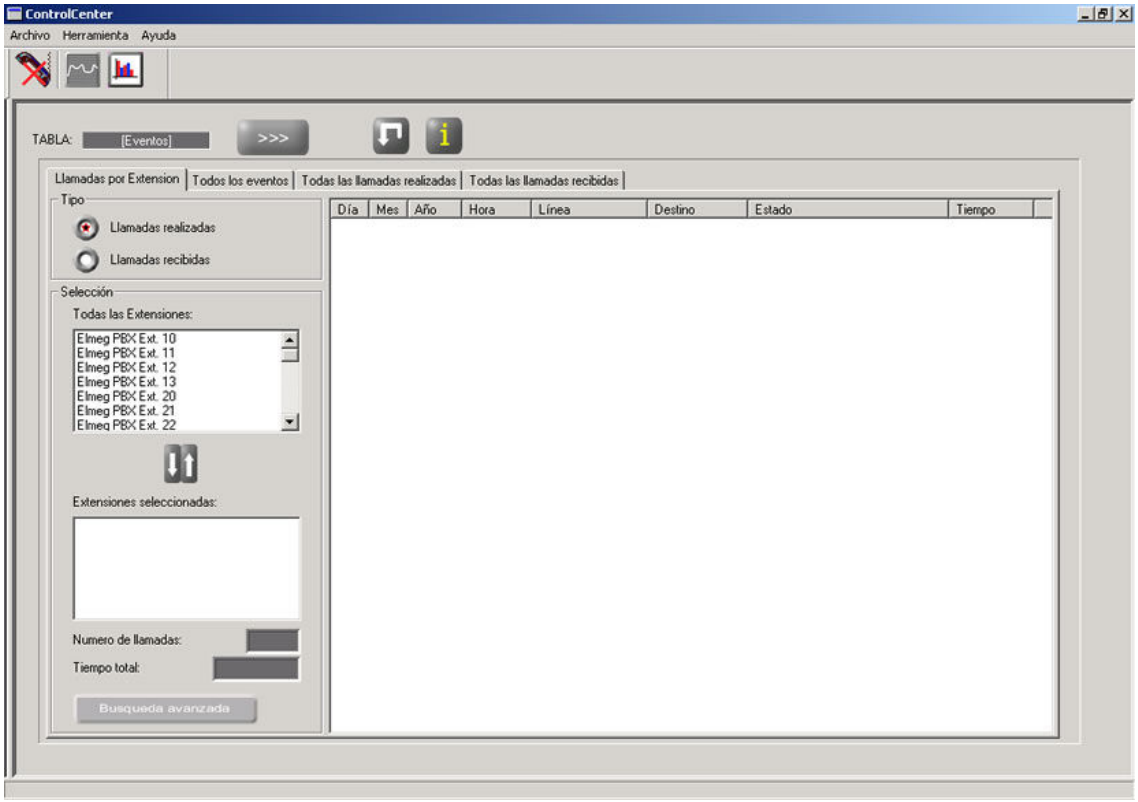
NOTA: Sea cual sea el camino a bloquear/desbloquear una extensión, la acción realizada se actualizará en el resto de componentes de la ventana principal.

3.5 Vista ‘Base de datos de llamadas’

En esta vista se puede acceder a la base de datos de todas las llamadas. Todos los eventos (cambios de estado en las extensiones) se guardan en la base de datos. Para visualizar el 100% de esta vista es necesario pulsar el boton de "Extender vista":



con lo que la ventana principal cambia mostrando únicamente la vista de base de datos de llamadas:



Esta vista está organizada en pestañas:

Todos los eventos

Muestra todos los eventos existentes en la base de datos.

Todas las llamadas realizadas

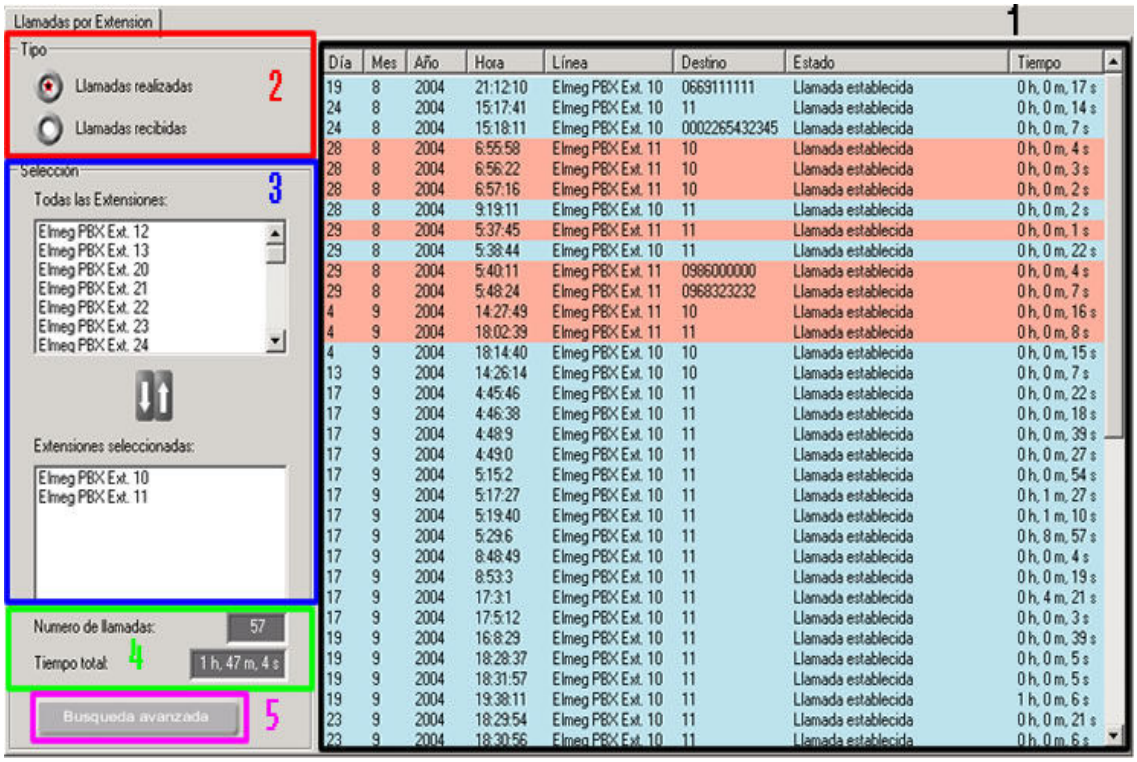
Muestra todas las llamadas realizadas existentes en la base de datos.

Todas las llamadas recibidas

Muestra todas las llamadas recibidas existentes en la base de datos.

Llamadas por Extensión

Esta pestaña permite hacer búsquedas personalizadas de llamadas en la base de datos.



1. Lista de llamadas encontradas

Muestra el resultado de la búsqueda resultado de aplicar todos los filtros.

2. Selección de tipo de llamadas

Permite filtrar la búsqueda solamente a llamadas realizadas o recibidas.

3. Selección de extensiones

En la lista superior se muestran todas las extensiones de la PBX. Con las flechas  seleccionamos o deseleccionamos las extensiones que serán objeto de las llamadas a buscar. Como vemos en la figura (que hay seleccionadas 2 extensiones), las llamadas encontradas correspondientes a cada extensión se muestran en colores diferentes para facilitar su distinción.

4. Información del resultado de la búsqueda

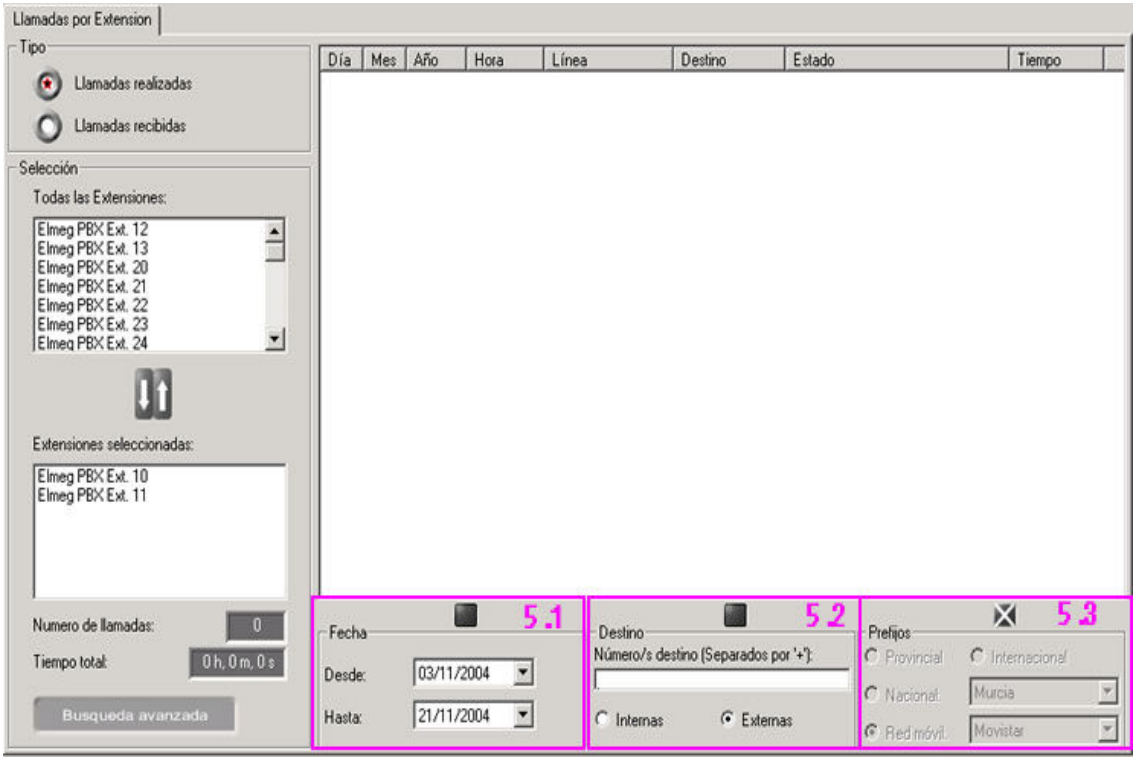
Este cuadro muestra información de las llamadas encontradas como resultado de la búsqueda. Esta información es:

- Número de llamadas encontradas
- Tiempo total de todas las llamadas encontradas

Esta información se recalcula cada vez que la lista de llamadas encontradas cambia.

5. Búsqueda avanzada

Es posible realizar búsquedas de llamadas avanzadas, es decir, imponiendo más condiciones de búsqueda o filtros. Si pulsamos el botón "Búsqueda avanzada" automáticamente se desplegará un panel con los posibles filtros:



5.1 Filtro de fecha

Permite acotar la búsqueda entre 2 fechas.

5.2 Filtro de Tipo de números destino / origen

Este filtro tiene 2 niveles. El primer nivel permite discriminar la búsqueda entre llamadas internas y externas. Una vez seleccionado el tipo de llamadas entre "Internas" o "Externas" se puede opcionalmente añadir un segundo nivel de búsqueda. Para ello en el campo de texto superior podemos añadir números de teléfono separados por el carácter '+' que:

- En el caso de estar seleccionado en la seccion 2 "Llamadas realizadas", se filtrarán las llamadas únicamente realizadas a dichos números.
- En el caso de estar seleccionado "Llamadas recibidas", se filtrarán las llamadas únicamente recibidas desde dichos números.

5.3 Filtro de prefijos

NOTA: Para poder habilitar este filtro es imprescindible tener habilitado el Filtro de Tipo de números destino / origen y seleccionada la opción "Externas".

Permite filtar las llamadas realizadas o recibidas (según el tipo seleccionado en la sección 2) según:

- El ámbito de llamada :
    - Provincial
    - Nacional
    - Internacional
    - Llamada a/desde red móvil

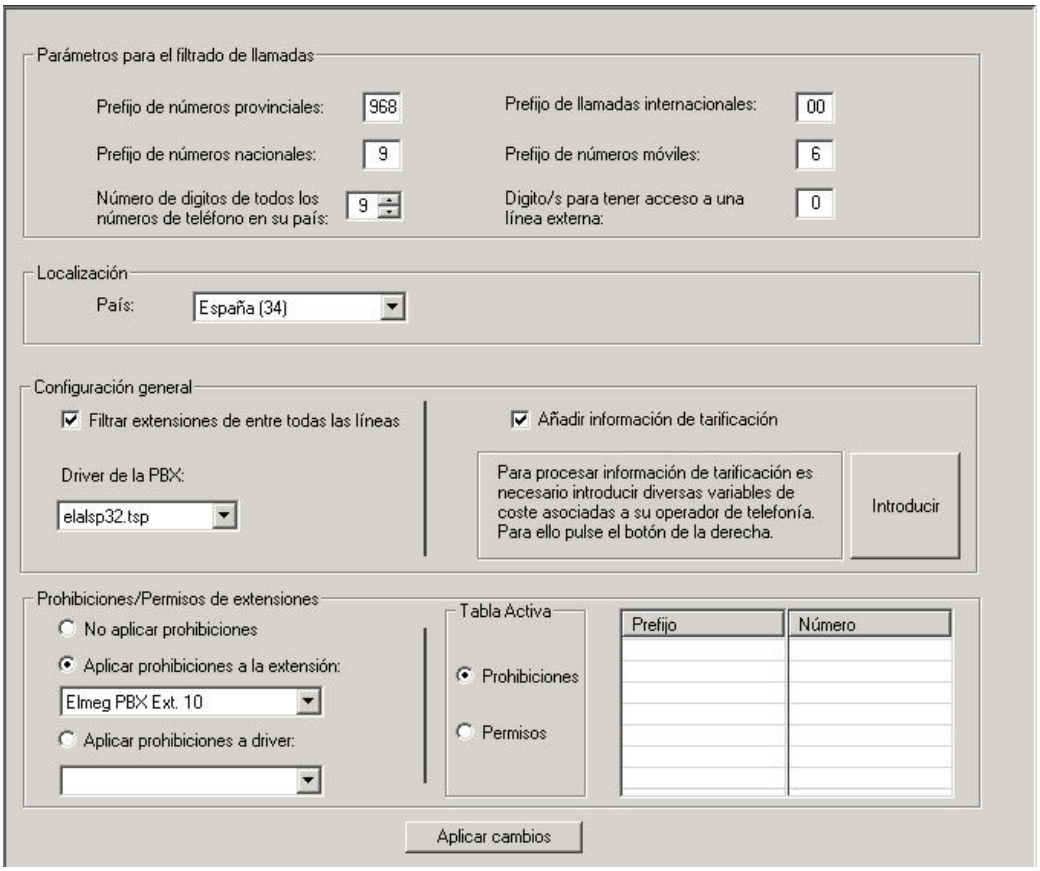
NOTAS:

1. En el caso de filtrar las llamadas por ámbito nacional es posible discriminar entre provincias.

2. En el caso de filtrar las llamadas por ámbito de red móvil es posible discriminar entre operadores de telefonía. Actualmente, en esta versión de PBX Control Center están disponibles los tres operadores de telefonía principales de España: Movistar, Vodafone y Amena.

3. En esta versión, para llamadas locales y nacionales únicamente se tiene en cuenta el operador Telefónica. En versiones posteriores se incluirá la posibilidad de definir nuevos operadores.

3.6 Vista 'Configuración'



En esta vista podemos configurar :

Parámetros para el filtrado de llamadas

Esta información es imprescindible para la vista 'Base de datos de llamadas'. A partir de ellos, es capaz de aplicar los filtros de ámbito de llamada, tipo de llamada y destino/origen. En esta versión de PBX Control Center sólo es posible tener un operador de telefonía fija, lo que limita las búsquedas de llamadas provinciales y nacionales. En una versión posterior se incluirá la posibilidad de definir varios prefijos para llamadas provinciales y/o nacionales.

Localización

Permite hubicar la aplicación en una zona geográfica determinada. Esta información es util a PBX Control Center para manejar información interna.

Configuración general

- **Filtar extensiones de entre todas las líneas**

Como ya se ha comentado anteriormente, PBX Control Center se comunica con la PBX a través del TSP o driver proporcionado por el fabricante e instalado en el sistema. En el sistema operativo, pueden existir otros TSPs instalados como por ejemplo, el que controla el Modem analógico. Cada TSP hace accesibles 1 o varias "líneas" al sistema operativo para que las aplicaciones puedan utilizarlas. Por ejemplo, el TSP de una PBX hace visibles a la aplicación las líneas de las extensiones, las líneas externas y otras posibles. PBX Control Center a priori negocia todas y cada una de las líneas visibles en el sistema, sean del TSP que sean. Ésto puede oscurecer su funcionamiento si solo se pretende monitorizar-administrar un dispositivo telefónico, como una PBX. Esta opción dentro de "Configuración general" permite indicarle a PBX Control Center que ignore todas las líneas que no pertenezcan al driver seleccionado.

¿Como conozco cuál es el driver de mi PBX?

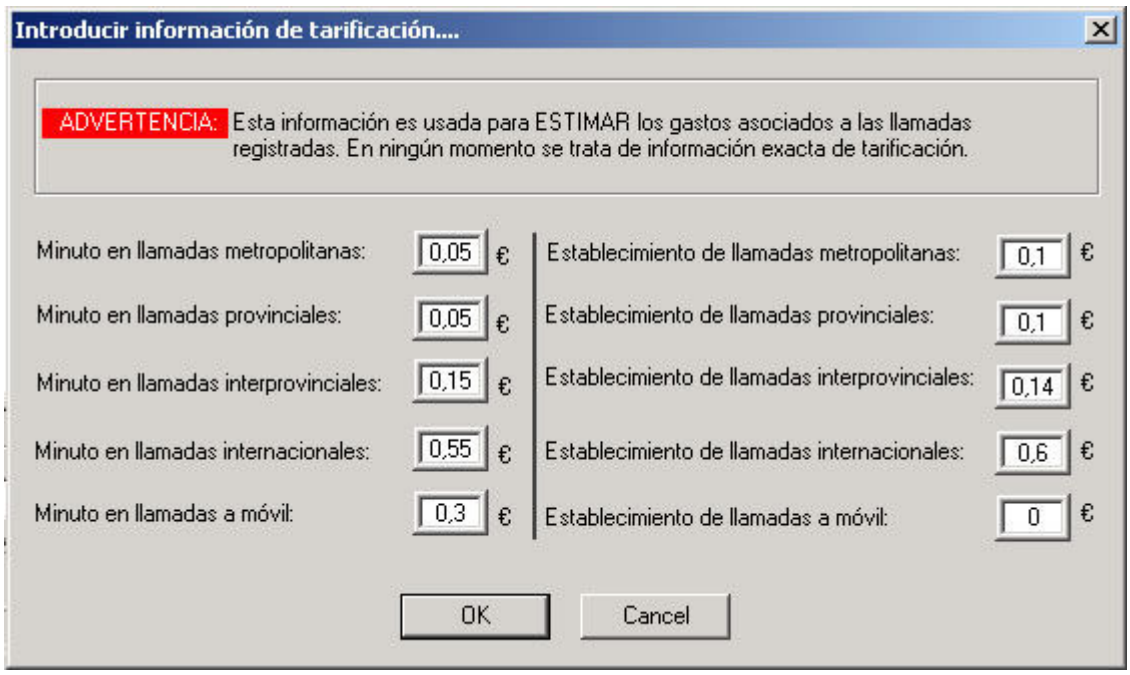
Existen 2 maneras:

1. Mirar en la documentación del TSP del fabricante o de la PBX.
2. Con el propio PBX Control Center. Estando en la vista de monitorización, debemos seleccionar en el cuadro Estado del sistema una extensión perteneciente a la PBX y automáticamente en el desplegable inferior se selecciona el driver asociado. Identificar una extensión que pertenezca a la PBX no supone mucha dificultad: el nombre (número de extensión) puede ser la clave. Si es imposible reconocer una extensión que pertenezca a una PBX en concreto podemos hacer lo siguiente. Generamos una llamada (simplemente con levantar el terminal de la extensión vale) desde una extensión de la PBX; la llamada aparecerá en la lista de monitorización, donde en la columna "Línea" aparecerá el nombre de la extensión.

Simplemente pinchando en ese registro de la lista el cuadro Estado del sistema se actualizará y podremos ver el TSP asociado.

- **Añadir información de tarificación**

A los informes generados por PBX Control Center es posible añadirles información de costes estimados de las llamadas realizadas. Para ello es necesario marcar esta opción y rellenar todos los campos de costes del cuadro de diálogo que aparece al pulsar el botón "Introducir".



Como indica el propio diálogo, los costes calculados no son en ningún momento información exacta de tarificación de la centralita. Es, simplemente una estimación de los mismos.

Prohibiciones/Permisos de extensiones

Permite definir prohibiciones en cuanto a realización de llamadas a las extensiones. Existen 2 formas de aplicar estas prohibiciones:

1. Individualmente a las líneas ó
2. definir las prohibiciones a un TSP, con lo que se aplicarán a todas las líneas que le pertenezcan.

En el primer caso, cada extensión tiene asociadas 2 tablas:

- Tabla de prohibiciones
- Tabla de permisos

Ambas tablas tienen 2 columnas, "Prefijos" y "Numeros".

En el segundo caso, es cada TSP (es decir, todas las extensiones asociadas a él) quien tiene asociadas esas 2 tablas.

Tabla prohibiciones

Define los prefijos y números prohibidos de esa extensión o TSP. La extensión que tenga esta **tabla como activa** no podrá realizar llamadas a ninguno de los números prohibidos ni a ningún número que comience por alguno de los prefijos prohibidos.

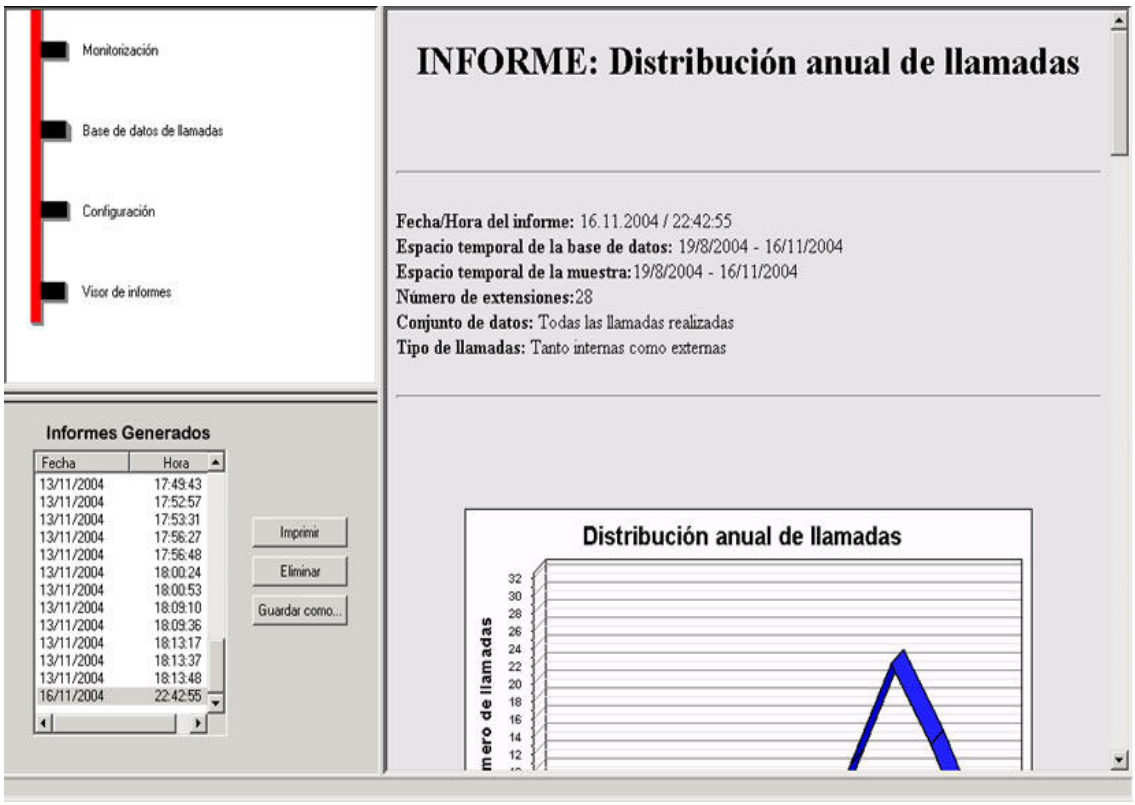
Tabla permisos

Define los prefijos y números permitidos de esa extensión o TSP. La extensión que tenga esta **tabla como activa** únicamente podrá realizar llamadas a los números permitidos o a los números que comiencen por alguno de los prefijos permitidos.

NOTA: Es necesario pulsar el botón 

para que los cambios tengan efecto. Además, si se modifica la opción "Filtrar extensiones de entre todas las líneas" es necesario reiniciar la aplicación ya que es necesario renegociar las extensiones que PBX Control Center monitorizará.

3.7 Vista 'Visor de informes'



Desde esta vista se puede consultar el archivo de informes generados por PBX Control Center. La generación de informes comparativos se lleva a cabo desde el cuadro de diálogo "Generación de informes estadísticos", pulsando en el botón  de la barra de herramientas estando como vista activa la de 'Base de datos de llamadas'.

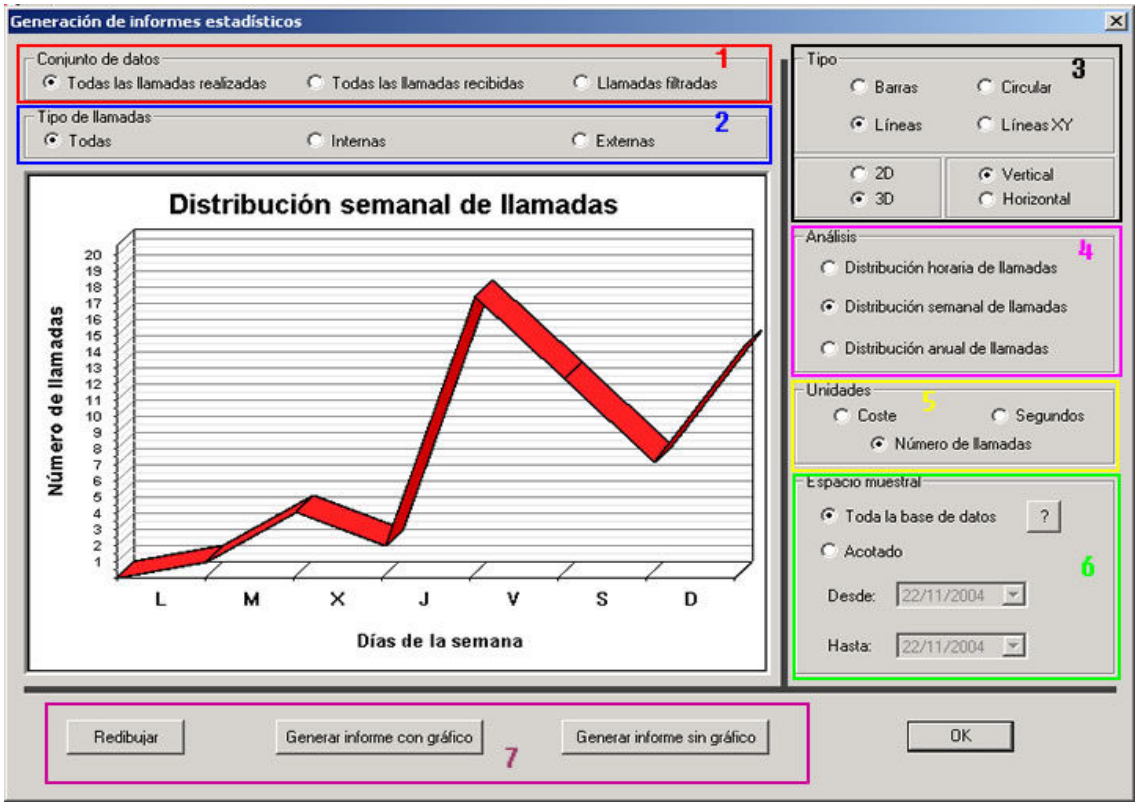
Desde el panel "Informes generados" es posible navegar entre los diferentes informes archivados así como realizar cualquiera otra de estas acciones:

- Imprimir el informe
- Guardarlo con otro nombre y/o otro destino.
- Eliminar el informe seleccionado

3.8 Generación de informes estadísticos

Teniendo la vista 'Base de datos de llamadas' como vista activa de la ventana principal, podemos acceder a la herramienta de generación de informes estadísticos del tráfico. El objetivo de esta utilidad es el de **auditar** la información del tráfico de llamadas de la centralita almacenado en la base de datos.

Pulsando en la barra de herramientas  :



1. Conjunto de datos

Acota el espacio muestral del análisis estadístico según los siguientes tipos de llamadas:

- Llamadas realizadas
- Llamadas recibidas

- Llamadas filtradas: el conjunto de llamadas filtradas es el resultado de una búsqueda personalizada en la vista 'Base de datos de llamadas'. Es decir, es el conjunto de llamadas encontradas que llenan la lista en el momento de iniciar la herramienta de generación de gráficos estadísticos.

2. Tipo de llamadas

Al igual que en el apartado anterior, acota el espacio muestral del análisis a llamadas:

- Internas: llamadas realizadas de extensión a extensión
- Externas: llamadas realizadas a la red pública.
- Todas: internas + externas.

3. Tipo de gráfico

Podemos personalizar el gráfico a generar en el análisis seleccionado. Podemos elegir entre diferentes tipos de gráficos:

- Barras
- Circular
- Líneas
- Líneas XY

y para cada uno elegir entre 2 Dimensiones o 3 (a excepción de Líneas XY que dispone solamente de la versión en 2D). Además se puede elegir la orientación entre vertical u horizontal.

4. Análisis estadístico

En esta sección se selecciona el análisis estadístico a realizar. En esta versión de PBX Control Center hay disponibles 3 análisis diferentes:

- **Distribución horaria de llamadas**: genera un gráfico donde se muestra la distribución de las llamadas (del conjunto de llamadas seleccionado) a lo largo del dia, en intervalos horarios de 2 horas. La utilidad de este análisis es encontrar fácil y visualmente el rango horario del dia en el que se realizan/reciben más llamadas. De este modo, para un administrador es posible configurar en la PBX la tabla de encaminamiento de llamadas , definiendo de forma eficiente los operadores de telefonía a usar en cada rango para, de esta manera, minimizar el gasto.

- **Distribución semanal de llamadas:** con la misma idea que en el caso anterior, genera un gráfico donde se muestra la distribución de las llamadas a lo largo de la semana.

- **Distribución anual de llamadas:** genera un gráfico donde se muestra la distribución de las llamadas a lo largo del año.

5. Unidades

Para todos los gráficos generados podemos definir las unidades del eje de abscisas (las unidades del eje de ordenadas están determinadas por el tipo de análisis). Se pueden elegir entre:

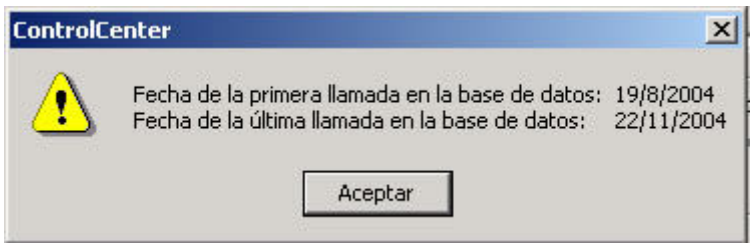
- Número de llamadas
- Número de segundos total de todas las llamadas
- Coste estimado (solo disponible para llamadas externas)

6. Espacio muestral

Otra forma de acotar el espacio muestral el cual analizaremos, es de forma temporal.

- Toda la base de datos: no acotamos el espacio muestral en términos temporales. Buscamos todas las llamadas de la base de datos atendiendo a

los demás criterios seleccionados. Si pulsamos sobre el botón  situado a la derecha se mostrará un cuadro de diálogo indicándonos el actual espacio temporal de la base de datos, es decir, la fechas de la primera y ultima llamadas registradas.



- Acotado: se definen las fechas de principio y fin de las muestras a analizar.

7. Opciones de generación del informe

El informe estadístico está compuesto por:

- el gráfico generado según el tipo de análisis escogido
- tabla comparativa de llamadas según la extensión
- (opcionalmente) tabla comparativa de costes estimados

Como opción, es posible generar el informe con gráfico o sin gráfico.

3.9 Formato del fichero de configuración

Fichero de configuración 'config.ini'

PBX Control Center utiliza un fichero .INI para almacenar información persistente de configuración. En principio, este fichero no debe ser editado de forma manual ya que su modificación corre a cargo de PBX Control Center. Sin embargo, a continuación se explican cada una de sus secciones por si en algún momento fuera necesario la modificación en caso de desintegridad del fichero. Este fichero debe estar localizado en la ruta donde se encuentra el ejecutable de la aplicación.

[DocColor]

Esta sección es parte de las secciones dedicadas a personalizar la apariencia de los informes estadísticos generados. En particular, define el color de fondo de los informes. Para ello define 3 parámetros:

- red=231 : cantidad de rojo en un esquema RGB
- green=227 : cantidad de verde en un esquema RGB
- blue=231 : cantidad de azul en un esquema RGB

[TableHeaderColor]

Define el color de fondo de la cabecera de las tablas comparativas generadas en los informes estadísticos. Parámetros:

- red=156 : cantidad de rojo en un esquema RGB
- green=158 : cantidad de verde en un esquema RGB
- blue=156 : cantidad de azul en un esquema RGB

[TableRowsColor]

Define el color de fondo de las filas de las tablas comparativas generadas en los informes estadísticos. Parámetros:

- red=214 : cantidad de rojo en un esquema RGB
- green=211 : cantidad de verde en un esquema RGB
- blue=214 : cantidad de azul en un esquema RGB

[ChartPosition]

Indica si el gráfico generado será mostrado antes de las tablas comparativas o después en el informe estadístico. Parámetro:

- chartPosition=0 : 0 si antes, 1 después.

[ProviderFilter]

Opción 'Filtrar extensiones de entre todas las líneas' de la vista de configuración. Parámetro:

- provider=8 : -1 si la opción no está seleccionada

  >0 indica el identificador del TSP o driver seleccionado en la lista desplegable.

[CostParams]

Opción 'Añadir información de tarificación' de la vista de configuración. Parámetros:

- costs=1 : 0, la opción no está seleccionada; 1, sí está seleccionada.
- minMetropolitan=0.05 : Coste promedio de un minuto en llamadas metropolitanas
- minProv=0.05 : Coste promedio de un minuto en llamadas provinciales
- minInterProv=0.15 : Coste promedio de un minuto en llamadas interprovinciales (nacionales)

- minIntern=0.55 : Coste promedio de un minuto en llamadas internacionales
- minMobile=0.3 : Coste promedio de un minuto en llamadas a móviles
- staMetropolitan=0.1 : Coste promedio de establecimiento de llamada en llamadas metropolitanas
- staProv=0.1 : Coste promedio de establecimiento de llamada en llamadas provinciales
- staInterProv=0.14 : Coste promedio de establecimiento de llamada en llamadas interprovinciales
- staIntern=0.6 : Coste promedio de establecimiento de llamada en llamadas internacionales
- staMobile=0 : Coste promedio de establecimiento de llamada en llamadas a móviles

[Prefixes]

Contenedor que define todos los parámetros necesarios para el filtrado de llamadas.

- provPrefix=968 : Prefijo de números provinciales
- natPrefix=9 : Prefijo de números nacionales
- numDigitsPrefix=9 : Número de dígitos de todos los números de teléfono en el país
- interPrefix=00 : Prefijo de llamadas internacionales
- mobilePrefix=6 : Prefijo de números móviles
- extPrefix=0 : Dígito/s para tener acceso a una línea externa

[ApplyTables]

Indica el modo de aplicar las tablas de prohibiciones/permisos: Parámetro:

- apply=1 :
    - 0, no se aplican las prohibiciones/permisos
    - 1, se aplican las prohibiciones/permisos a cada extensión
    - 2, se aplican las prohibiciones/permisos al driver (a todas las extensiones del driver seleccionado)

A continuación, figurarán tantas secciones como extensiones esté monitorizando PBX Control Center, con el siguiente formato:

[Nombre extensión]

Define las tablas de números permitidos y números prohibidos asociados a esa extensión. Parámetros:

- NumAllowed= : Cadena de números permitidos
- NumDenied= : Cadena de números prohibidos
- PrefAllowed= : Cadena de prefijos permitidos
- PrefDenied= : Cadena de prefijos prohibidos
- TableActive= : Tabla activa. 1, números prohibidos. 0, números permitidos.

La cadenas tienen el siguiente formato:

número1/número2/...../númeroN

Y por último figurarán también tantas secciones como drivers instalados haya en el sistema operativo. El formato es similar al caso anterior:

[NombreDelDriver.tsp]

Define las tablas de números permitidos y prohibidos asociadas al conjunto de líneas pertenecientes a ese driver. Parámetros:

- DeviceID= : Número de dispositvo del driver. Cada driver TAPI tiene un único DeviceID en el sistema.
  NumAllowed= : cadena de números permitidos
  NumDenied= : cadena de números prohibidos
  PrefAllowed= : cadena de prefijos permitidos
  PrefDenied= : cadena de prefijos prohibidos
  TableActive= : Tabla activa. 1, números prohibidos. 0, números permitidos.

El formato de las cadenas de números es igual que en el caso de las extensiones.

Capítulo 4
Desarrollo de la herramienta

4.1 Introducción

En este capítulo se hará un profunda descripción del marco de trabajo utilizado para el desarrollo de la aplicación PBX Control Center, entendiendo por tal, desde el lenguaje de programación elegido hasta las librerías y bibliotecas utilizadas. En segundo lugar se justificarán algunas decisiones tomadas en el diseño de la aplicación y en la resolución de algunos problemas concretos, como el acceso a base de datos, generación de informes en HTML (*HyperText Markup Language*) o la generación de la ayuda de la aplicación en formato HTML comprimido (ayuda estándar de Windows). Se abordarán éstas y otras cuestiones desde el punto de vista de justificar la decisión tomada frente a otras alternativas sin entrar en ningún momento en detalles de la implementación ni comentar el código fuente de la aplicación ya que se sale de los objetivos de este texto.

Finalmente, se presentará el escenario utilizado en las pruebas realizadas detallando el hardware utilizado y los problemas que podemos encontrarnos.

4.2 Objetivos iniciales

La funcionalidad de la herramienta desarrollada no estaba nada clara cuando se establecieron los objetivos iniciales del proyecto. Esto se debía al casi total desconocimiento de la tecnología CTI y de las librerías disponibles para tal fin. Era también desconocido el lenguaje de programación a usar así como las características requeridas de la centralita disponible en un primer momento, una Alcatel OmniPCX Office. Por todos estos motivos, en una primera valoración se fijó como objetivo el desarrollar una aplicación no necesariamente con entorno gráfico que hiciera algún tipo de control de control o registro de las llamadas que se producían en la centralita.

4.3 Estándar CTI utilizado

En capítulos anteriores se han contemplado los posibles estándares CTI existentes en la actualidad barajándose sus ventajas e inconvenientes. Este proyecto se titula "*Diseño e implementación de una aplicación CTI basada en el estándar TAPI para el control de centralitas de conmutación telefónica"* por lo que desde un primer momento el estándar TAPI de Microsoft fue la opción predominante. Sin embargo, se han barajado otras alternativas a TAPI como son:

- **JTAPI**

La posibilidad de escribir aplicaciones de telefonía en Java reduce notablemente la complejidad del programador gracias a las grandes características de este lenguaje de programación como son: la facilidad de uso, el garbage collector, la orientación a objetos, su modelo de manejo de eventos, callbacks mediante la Reflection API, etc. En el desarrollo de aplicaciones de telefonía la portabilidad de Java no juega tan importante papel como en otros campos del software. Esto es así debido a que la implementación de JTAPI es una mera "wrapper layer" (capa envoltorio) de las librerías TAPI. Es decir, JTAPI delega las llamadas a los métodos en llamadas a código nativo C de las librerías TAPI, quienes se comunican con el TSP del dispositivo. Por lo tanto, esta plataforma está ligada al estándar TAPI y por consiguiente, al sistema operativo Windows. Esto, unido a la falta popularidad de la plataforma y por consiguiente, una reducida comunidad de desarrolladores en torno a ella que proporcione recursos al programador, provocó que se abandonara esta opción para este proyecto.

- **TSAPI**

El estándar TSAPI de Novell, como ya se ha comentado en capítulos anteriores, proporciona una gran potencia a las aplicaciones ya que especifica un gran conjunto de funciones para un gran amplio rango de dispositivos. Aunque

en las últimas versiones, esta plataforma soporta comunicación sobre protocolos distintos a IPX y está disponible para un gran abanico de plataformas software (Windows, Novell Netware, HP-UX…), este estándar no es implementado por un gran número de modelos de centralitas. Concretamente, la PBX disponible para las pruebas no implementa este estándar por lo que esta opción fue totalmente descartada.

Finalmente TAPI fue la solución elegida entre ella y JTAPI. Además de las razones expuestas anteriormente, la amplia comunidad formada entorno a TAPI y los recursos disponibles en Internet hacen de esta plataforma idónea para el desarrollo de aplicaciones CTI.

4.4 Entorno de desarrollo

La elección del lenguaje de programación está fuertemente condicionada por el estándar CTI utilizado. La versión de TAPI usada también puede abrir el abanico de lenguajes a utilizar ya que como se ha comentado en capítulos anteriores, la versión 3.x está implementada basada en objetos COM lo que posibilita el acceso desde casi cualquier lenguaje.

S.O.

Microsoft Windows 2000 Profesional.

Versión TAPI

La versión utilizada por la aplicación de TAPI depende de la versión de Windows utilizada, de la implementación de TAPI que lleva la centralita y de la versión que implemente el TSP. Al final, la versión utilizada por la aplicación es el mínimo de estas tres variables. En el escenario utilizado para las pruebas:

- Versión de la implementación de la PBX: 2.0
- Versión del TSP: 2.0
- Versión del sistema operativo Windows 2000: 2.2

Por lo que la versión que utiliza la herramienta desarrollada (PBX Control Center) es la 2.0. El SDK de la versión 2.0 del estándar TAPI está basado en C, por lo que éste es el lenguaje, a priori, a utilizar.

Framework C++

A pesar de ser el lenguaje C el lenguaje sobre el que está basado el SDK de la versión 2.0 y por lo tanto, las llamadas a las funciones de la librería TAPI, el lenguaje utilizado ha sido C++ haciendo uso para ello de un framework de clases construido a partir de las llamadas del SDK y obtenido con el libro "*Windows Telephony Programming. A developer's Guide to TAPI"* [7]. Este framework permite desarrollar aplicaciones de telefonía ayudándose de la programación orientada a objetos, modelando con cierta realidad cada elemento que forma parte en la comunicación telefónica a través de las clases. Así mismo, el manejo de eventos se modela desde el punto de vista de la herencia y el polimorfismo y el manejo (reserva) de memoria, aspecto crítico en el aprendizaje de TAPI, es parcialmente ocultado al programador. Todos estos aspectos hacen a las aplicaciones desarrolladas con este framework, más fiables, extensibles y depurables.

IDE

El entorno integrado de desarrollo (IDE) utilizado es Microsoft Visual C++ 6.0 Profesional. La elección de éste frente a otros entornos OpenSource como *Bloodshed Dev-C++* responde a la integración completa con las bibliotecas MFC usadas para desarrollar el interfaz gráfico.

4.5 Librerías y herramientas utilizadas

A continuación se describen las librerías, SDK y componentes software "third-party" utilizados para el desarrollo de PBX Control Center:

- **Microsoft Platform SDK** [6]

Es la plataforma de desarrollo (SDK) para la programación de aplicaciones en el sistema operativo Windows. Contiene todos las ficheros y librerías de desarrollo para la programación en este sistema así como una completa documentación de los API y pequeñas herramientas que hacen la vida más fácil al programador. Además contiene numerosos ejemplos con el código fuente incluido. La versión utilizada en este proyecto es la *release* de Febrero del 2003.

Es necesario configurar adecuadamente el entorno de Visual C++ para que encuentre los ficheros de cabecera y librerías de desarrollo de TAPI. Para ello, desde el menú *Tools* | *Options*... añadimos a la lista de directorios la ruta a los ficheros de cabecera del Platform SDK.

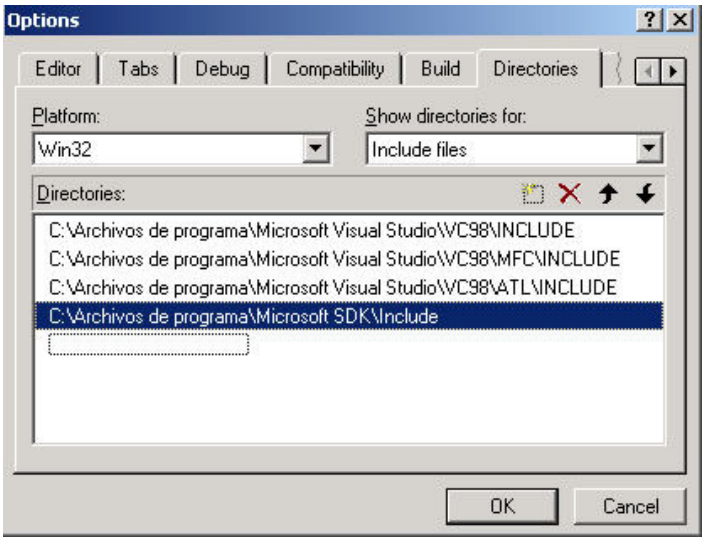


Figura 16. Configuración de cabeceras

Análogamente, añadimos a la lista de ficheros de librerías la ruta a las librerías del Platform SDK:

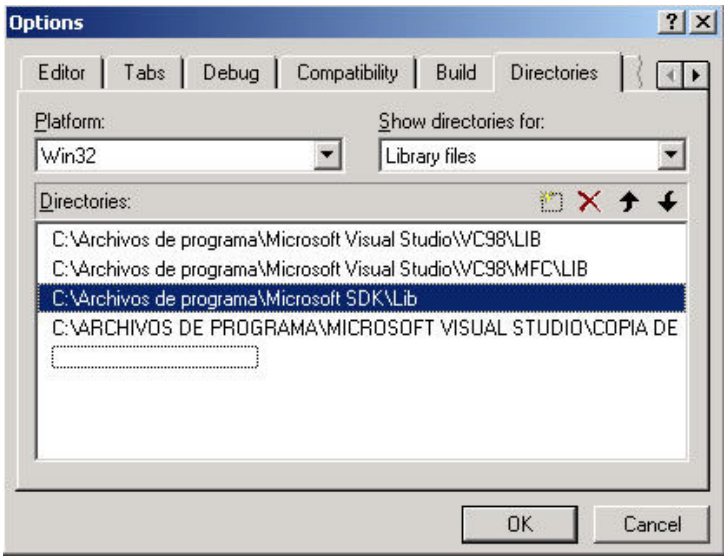


Figura 17. Configuración de librerias

- **MFC (Microsoft Foundation Classes)**

Las MFC son una enorme colección de clases C++ desarrolladas por Microsoft para permitir la programación de aplicaciones Windows en el lenguaje C++. Son más que una colección de clases, es un framework de desarrollo que proporciona además de las clases, una especificación de definición de componentes gráficos, un modelo de gestión de eventos y una integración de todos estos elementos en el desarrollo de interfaces gráficas. Estas librerías vienen integradas con el entorno Microsoft Visual C++ 6.0.

- **The Telephony Framework**

Es el framework de clases C++ para la programación de aplicaciones TAPI descrito en el apartado anterior, obtenido con el libro "*Windows Telephony Programming. A developer's Guide to TAPI"* escrito por Chris Sells.

- **Shell LightWeigth API**

Es una librería dentro del Platform SDK de Microsoft que contiene funciones de manejo de ficheros y directorios. En el desarrollo de PBX Control Center se ha enlazado a esta librería de forma estática, de forma que no sea necesario

adjuntar la DLL con el programa. Para ello, es necesario configurar el entorno de Visual C++ en las opciones del proyecto, en el menú *Project* | *Settings*...

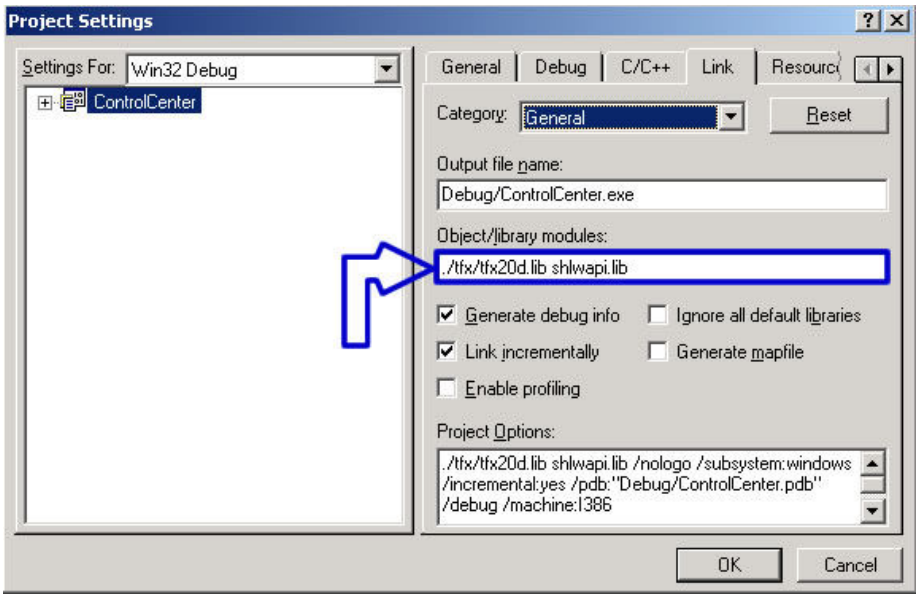


Figura 18. Configuración de módulos

Vemos que además de shlwapi.lib se encuentra la librería correspondiente al "The Telephony Framework" descrito anteriormente.

- **Mozilla ActiveX Control**

Es el motor del navegador web Mozilla implementado como un control ActiveX de Microsoft. De esta manera y como cualquier control ActiveX es posible incrustarlo fácilmente en cualquier aplicación Windows. Es utilizado por PBX Control Center para visualizar informes HTML generados por la propia aplicación sin tener que recurrir a un navegador externo. Más información de Mozilla ActiveX Control en la página web del proyecto: http://www.iol.ie/~locka/mozilla/mozilla.htm

- **CGraph**

Un conjunto de clases para la representación de gráficos estadísticos desarrolladas por *Brian Convery* y obtenidas en http://www.codeguru.com.

4.6 Algunos aspectos de diseño

4.6.1 Acceso a base de datos

PBX Control Center almacena en una base de datos todas las llamadas monitorizadas permitiendo a posteriori realizar operaciones de búsqueda de llamadas y auditoría del tráfico de la centralita realizando análisis estadísticos y estimaciones de costes.

Frente a ésta necesidad de almacenamiento y acceso a base de datos se plantearon dos soluciones:

- **Utilización de un servidor de bases de datos como MySQL**

Sin duda, es la mejor opción de entre todas las posibles. Usar un servidor de bases de datos (open source) como MySQL o PostgreSQL proporciona alta escalabilidad, robustez de la información, rapidez en las operaciones y gran flexibilidad. Otra de las grandes ventajas de esta opción es poder tener la base de datos en una máquina diferente a la que ejecuta la aplicación.

En contra, esta opción "obliga" al administrador de la aplicación a instalar un servidor de bases de datos y crear las tablas necesarias así como configurar la aplicación adecuadamente para que la localice.

- **Utilizar almacenamiento local en ficheros**

Es la opción utilizada por la gran mayoría de aplicaciones comerciales. Todas ellas guardan la información en ficheros con un formato propietario. Esta opción proporciona mayor control sobre la información y facilidad de uso de la aplicación ya que es ella quien gestiona todo el almacenamiento de la información no dependiendo de servidores externos. Sin embargo tiene dos claras desventajas:

1. El diseño del formato de los ficheros así como el tener implementar rutinas eficientes de almacenamiento / lectura puede llevar mucho tiempo.
2. Las rutinas implementadas nunca llegarán a tener la misma eficiencia que un servidor de bases de datos y si la información almacenada es muy grande, puede llevar a grandes retardos en la búsqueda de información ya que la información es guardada de forma secuencial.

Finalmente la opción elegida es un híbrido entre las dos alternativas presentadas anteriormente. Lo que se intenta es aprovechar todas las características del almacenamiento en una base de datos (indexamiento, robustez, rapidez, etc) evitando el tener que disponer de un servidor externo. Para ello se ha utilizado una base de datos local Microsoft Access que permite la escritura/lectura en un fichero local a través de un driver tal y como si se estuviera accediendo a un servidor de bases de datos, realizando consultas SQL. A continuación se detalla el proceso de despliegue de dicha base de datos:

Microsoft Access y ODBC (Open DataBase Connectivity)

Es un estándar que define el acceso a bases de datos desarrollado por el *SQL Access group* en 1992. El objetivo de ODBC es hacer posible el acceso a cualquier base de datos desde cualquier aplicación sin importar cuál sea la base de datos. En otras palabras, cada base de datos está implementada de forma diferente a las demás y el acceso a los datos puede también implementarse de forma diferente o mediante varios lenguajes como SQL. Normalmente, para manejar una base de datos en concreto tendríamos que conocer el lenguaje o SDK que utiliza. Lo que consigue ODBC es poder escribir una aplicación que interactúe con una base de datos y que podamos cambiar la base de datos usada sin tener que modificar una sola línea del programa. Para tal fin, ODBC introduce una capa de abstracción entre la aplicación y el driver que maneja la base de datos siguiendo la WOSA (*Windows Open Services Architecture*), es decir, con la misma arquitectura que lo hace TAPI. Es decir, la aplicación nunca

interactuará directamente con el driver de la base de datos sino que hará uso de ODBC quien delegará las llamadas en operaciones concretas del driver.

PBX Control Center hace uso de ODBC para la escritura/lectura de la base de datos Microsoft Access utilizada. Para usar ODBC, Windows dispone en el Panel de Control de un administrador de orígenes de datos ODBC. Un origen de datos (data source) es un conjunto de información (que se define en el sistema ODBC) sobre el acceso a la base de datos y que permite a las aplicaciones acceder de modo transparente a las bases de datos. El nombre del origen de datos será la información que necesite la aplicación para hacer uso de una base de datos u otra. Es imprescindible que el origen de datos esté configurado adecuadamente en el administrador de orígenes de datos del Panel de control; de otro modo, la aplicación será incapaz de localizar la base de datos. El origen de datos es configurado automáticamente en el proceso de instalación de PBX Control Center y el fichero de la base de datos **callreg.mdb** copiado en la ruta adecuada, por lo que no será necesario la configuración por parte del usuario. Sin embargo si modificamos posteriormente la ruta por defecto al fichero de la base de datos la aplicación dará error. Por este motivo se incluye un anexo una descripción de cómo crear estos orígenes de datos por si hubiera que modificarlo.

4.6.2 Generación de informes

Como se ha visto en el capítulo anterior, PBX Control Center incluye la posibilidad de generar informes comparativos de las llamadas registradas incluyendo información adicional de costes estimados. Para el formato de estos informes se barajaron varias alternativas:

- Formato propietario
- XML
- HTML

El formato propietario tiene las mismas ventajas y desventajas analizadas en el apartado anterior. XML es un estándar para el almacenamiento de la información de gran popularidad aunque su principal desventaja es que posteriormente para la representación de los informes sería necesario en la aplicación un parser XML. HTML sin embargo es entendido por todos los navegadores Web. Por este motivo se ha elegido este formato para los informes ya que ofrecen la posibilidad de visualizarlos en cualquier browser y poder modificarlos fácilmente.

Además, PBX Control Center lleva incrustado como visor de informes un control ActiveX del motor de renderizado de Mozilla (Gecko) para poder visualizar los informes desde el propio programa sin recurrir a navegadores externos.

4.6.3 Tarificación

PBX Control Center incluye en los informes información de tarificación estimada resultado de multiplicar el número de llamadas salientes por el precio introducido por el usuario en la parte de configuración de la aplicación. Aunque hace una clara distinción entre llamadas locales, interprovincipales, internacionales y llamadas a móviles ésta división no es aplicable a diferentes operadores ni diferentes paises, por lo que esta información solo puede tener carácter orientativo. El estandar TAPI especifica varias formas de recibir información de tarificación de la centralita (para ser más exactos, del dispositivo telefónico) siempre y cuando esta información sea enviada por la implementación de TAPI de la centralita. En el Apéndice B trata sobre cómo se obtendría dicha información a través de TAPI.

4.6.4 Ayuda de la aplicación

La ayuda de la aplicación se ha escrito en HTML integrándola luego en un único archivo de ayuda de Windows .chm, también llamado HTML comprimido. En el Apéndice C se muestra cómo se realizan éste tipo de ficheros.

4.7 Escenario de las pruebas

En un primer momento se disponía para las pruebas de una centralita Alcatel OmniPCX Office Rack 1. Esta central viene bloqueada de fábrica impidiendo la conexión desde el PC de una aplicación CTI distinta a las vendidas por Alcatel. Para poder desarrollar aplicaciones para esta centralita es necesario pagar una licencia que la desbloquee. Esta licencia es particular a cada centralita ya que se genera a partir del número serie de la CPU interna. Finalmente, ha sido posible realizar las necesarias pruebas mediante la adquisición de una central Elmeg D@vos 44.dsl de gama baja pero que implementa el estándar TAPI y no requiere el pago de ninguna licencia.

La centralita Elmeg se distribuye con el TSP que es capaz de establecer el enlace CTI sobre TCP/IP o mediante conexión USB. En las pruebas realizadas, la conexión estaba basada en TCP/IP.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

Se ha desarrollado una aplicación CTI capaz de operar con todas las centralitas que implementen el estándar TAPI cumpliendo no solo los objetivos marcados al comienzo del proyecto sino extendiendo las funcionalidades de la aplicación según iban surgiendo nuevas ideas. En el capítulo 4, Manual de usuario de PBX Control Center se detallan todas estas funcionalidades. Como resumen, a continuación se enumeran las principales funcionalidades implementadas:

- Monitorización de llamadas en tiempo real
- Posibilidad de colgar una llamada en curso
- Función de bloqueo de extensiones, entendiendo el bloqueo como la imposibilidad de realizar ni recibir llamadas.
- Definición de tablas de números prohibidos y números permitidos asociados a cada extensión o por grupos.
- Registro en base de datos de todas las llamadas
- Posibilidad de hacer búsquedas avanzadas de llamadas en la base de datos mediante un detallado formulario.
- Generación de informes de llamadas
- Generación de gráficos comparativos de llamadas siendo posible seleccionar entre diferentes tipos de gráficos y personalizarlos como se desee.
- Visor de los informes generados desde la aplicación

Para el desarrollo de esta herramienta el alumno ha tenido que aprender:

- El lenguaje de programación C++
- La especificación TAPI o parte de ella
- Dominar el framework C++ "The Telephony Framework"
- Programación con Microsoft Visual C++
- El API Win32
- La biblioteca de clases MFC

Finalmente comentar que la programación en Windows requiere larga curva de aprendizaje si se programa directamente usando el API Win32, es decir, en C puro. Sin embargo, eligiendo como lenguaje de programación C++ y usando las bibliotecas MFC se pueden llegar a crear interfaces gráficos complejos en relativamente poco tiempo. Esta misma consideración puede trasladarse a TAPI. Programar directamente usando el API en C de TAPI puede llegar a ser complicado por el hecho del uso de estructuras variables que se rellenan cuando se hacen llamadas a algunas funciones TAPI. Este tipo de estructuras y el manejo (reserva) de memoria que ello implica puede llevar a la implementación de programas inestables y con "memory leaks" si no se realiza de una forma adecuada.

5.2 Líneas futuras

Como se ha comentado anteriormente, en el desarrollo de la herramienta se han ido implementado nuevas funcionalidades según iban surgiendo nuevas ideas. A pesar de esto, han quedado en el tintero muchas ideas que por poner un punto y a parte en este proyecto se han aparcado. A continuación se listan algunas de estas ideas y carencias que tiene esta versión de PBX Control Center con la idea de poder ampliarse y seguir trabajando en nuevas versiones:

- Poder configurar la aplicación para que use un servidor de bases de datos externo como MySQL u Oracle introduciendo información como la dirección IP, el puerto y el nombre del origen de datos definido en el

sistema ODBC. Todo esto como alternativa al almacenamiento en base de datos local Microsoft Access implementado en esta versión. El uso del estándar ODBC hace que se pueda incluir esta alternativa sin modificar ninguna línea del manejo de la base de datos.

- Internacionalizar la aplicación, es decir, incluir soporte multi-idioma.
- Incluir módulo que trate los eventos de tarificación enviados por la centralita. Estos eventos no siempre se envían, depende de la centralita y del driver por lo que hay que prever este hecho adecuadamente.
- Poder configurar límites de gasto para las extensiones.
- Implementar algún tipo de encaminamiento de llamadas como una tabla ARS.
- Implementar un mecanismo ACR (*Automatic Call Routing*) que encamine las llamadas entrantes a determinadas extensiones siguiendo algún tipo de parámetro como tiempo de inactividad.
- Incluir soporte de varios operadores como ONO, Jazztel, etc. En esta versión de la herramienta sólo se tiene en cuenta el operador Telefónica. Para esta funcionalidad sería conveniente incluir una amplia base de datos de costes de llamadas y operadores. No se ha incluido esta funcionalidad en esta versión ya que esta base de datos debe ser actualizada cada cierto tiempo, siendo esta información difícil de obtener.

Apéndice A.
Configuración de un origen de datos ODBC

Abrir el gestor "Orígenes de datos (ODBC)" situado en el panel de control o (en sistemas operativos Windows 2000 o Windows Server) en Panel de control | Herramientas administrativas | Orígenes de datos (ODBC):

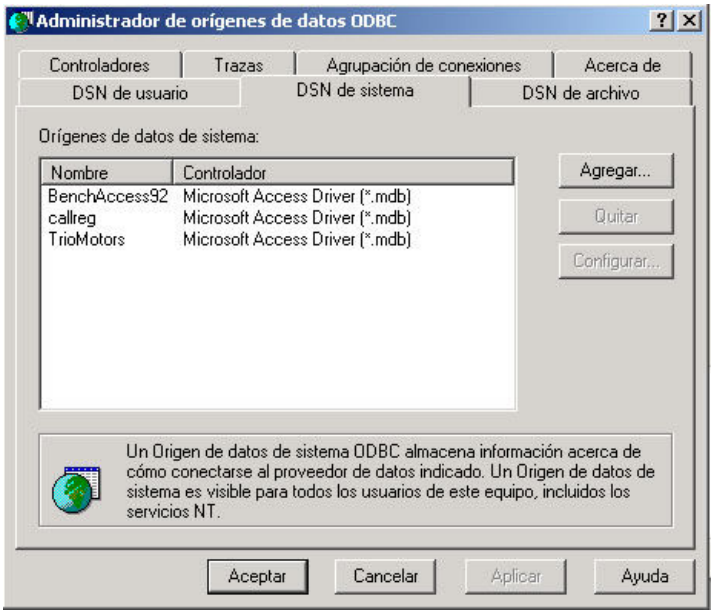


Figura 19. Administrador ODBC

Seleccionamos la pestaña DSN de sistema y pulsamos el botón "Agregar…" donde nos aparecerá un cuadro de dialogo con una lista de los drivers ODBC instalados en el sistema. Para utilizar una base de datos local Microsoft Access tendríamos que seleccionar "Microsoft Access Driver (*.mdb)".

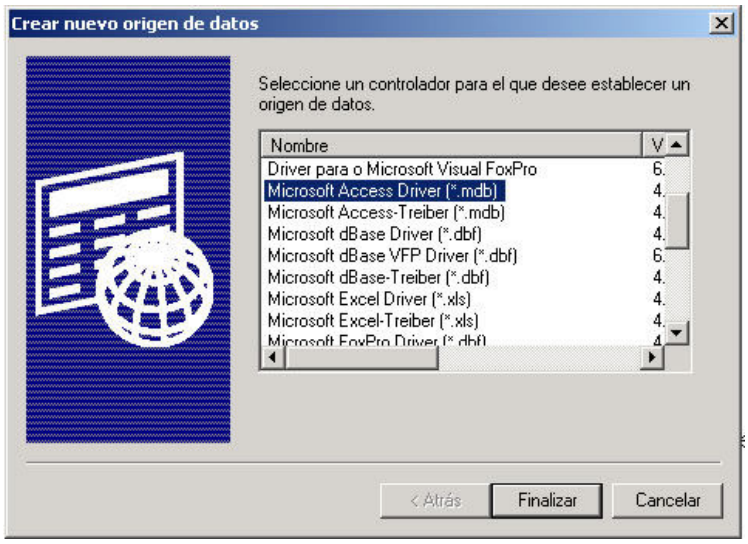


Figura 20. Nuevo origen de datos

Pulsamos finalizar y nos aparecerá un nuevo cuadro de diálogo en donde especificaremos el nombre del origen de datos (utilizado luego desde las aplicaciones), una descripción opcional y podremos seleccionar el fichero de .mdb de la base de datos a utilizar:

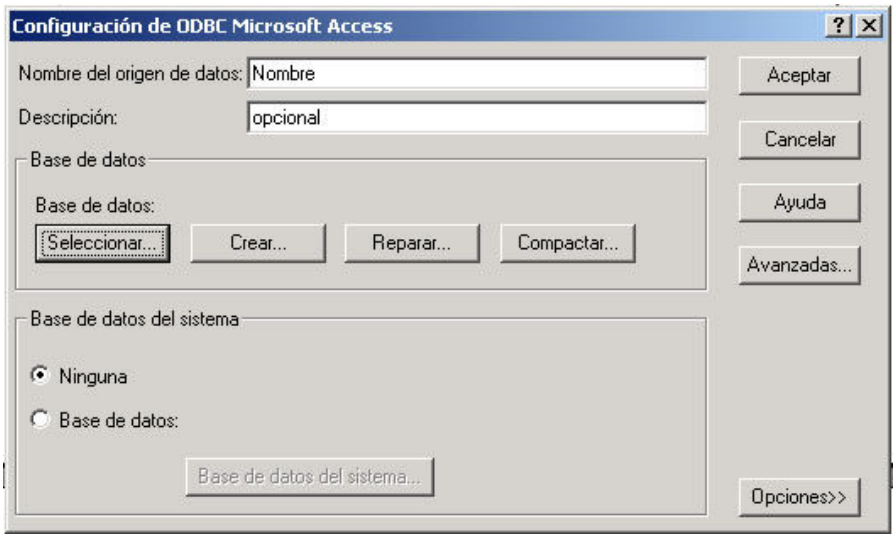


Figura 21. Configuración del origen de datos

Para ello, seleccionaremos una base de datos ya creada o podremos crear una nueva mediante los botones “Seleccionar…” o “Crear…”. Una vez hecho esto pulsamos “Aceptar” y el origen de datos queda registrado en el sistema.

Apéndice B. Tarificación en el estándar TAPI

En el estándar se especifican 3 mecanismos para acceder a la información de tarificación que puede enviar el dispositivo telefónico. Sea cual sea el método, tanto el TSP como la implementación de la centralita deben ser capaces de enviar tal información:

- Mediante los campos *dwChargingInfoOffset* y *dwCharginInfoSize* de la estructura LINECALLINFO. Esta estructura de tamaño variable es rellenada al llamar a la función **lineGetCallInfo.**

- Mediante un mensaje LINE_CALLINFO enviado desde el dispositivo telefónico con el flag *LINECALLINFOSTATE_CHARGINGINFO* indicando que la información de tarificación de la llamada asociada ha cambiado.

- Mediante extensiones del estandar TAPI. Un TSP de un dispositivo telefónico puede extender la especificación TAPI si es necesario definiendo nuevas funciones y/o estructuras, todas ellas específicas a él. Para utilizar estas extensiones específicas es necesario acudir al manual del TSP ya que no son estándar.

A continuación se muestra el fichero de log de la herramienta TB20 muy útil en la resolución de problemas con los TSP ya que permite hacer llamadas a TAPI y ver el resultado de las estructuras en memoria como si de un debugador se

tratase. El fichero de log se corresponde con el TSP Siemens HiPath TAPI 170 de la PBX Siemens HiPath 3000:

```
     ----------------------------------------------------TB20.log------
            ---------------------------------------------


3:48.55.289 : Calling lineInitializeEx
>     lphLineApp=x232508
>     hInstance=x1000000
>     lpfnCallback=x10158d7
>     lpszFriendlyAppName=x6f66c
>     lpdwNumDevs=x6f644
>     lpdwAPIVersion=x6f6d8
>       ->dwAPIVersion=x20002
>     lpInitExParams=x6f62c
>       ->dwOptions=x3
> 3:48.55.299 : lineInitializeEx returned SUCCESS
>   num line devs = 40
> 3:49.10.291 : Calling lineOpen
>     hLineApp=x800003ee
>     dwDeviceID=x13
>     lphLine=x232520
>     APIVersion=x20002
>     dwExtVersion=x0
>     dwCallbackInstance=x0
>     dwPrivileges=x6
>     dwMediaModes=x4
>     lpCallParams=x0
> 3:49.10.391 : lineOpen returned SUCCESS
>
> 3:50.14.243 : Calling lineMakeCall
>     hLine=x10399
>     lphCall=x2325d0
>     lpszDestAddress=x6f5e4
>     dwCountryCode=x0
>     lpCallParams=x0
> 3:50.14.243 : lineMakeCall returned x10344
> 3:50.14.553 : received LINE_REPLY
>   device=x0
```

```
>   cbInst=x0
>   param1=x10344,
>   param2=x0,
>   param3=x10322,
> 3:50.14.563 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x8, DIALTONE
>   param2=x0,
>   param3=x0,
> 3:50.14.563 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x40, CALLID
>   param2=x0,
>   param3=x0,
> 3:50.14.683 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x10, DIALING
>   param2=x0,
>   param3=x0,
> 3:50.14.683 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x8000, CALLERID
>   param2=x0,
>   param3=x0,
> 3:50.14.683 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x4000, TRUNK
>   param2=x0,
>   param3=x0,
> 3:50.15.715 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x200, PROCEEDING
>   param2=x0,
>   param3=x0,
> 3:50.15.715 : received LINE_CALLINFO
```

```
>   device=x10322
>   cbInst=x0
>   param1=x8000, CALLERID
>   param2=x0,
>   param3=x0,
> 3:50.15.715 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x10000, CALLEDID
>   param2=x0,
>   param3=x0,
> 3:50.15.735 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x20, RINGBACK
>   param2=x0,
>   param3=x0,
> 3:50.15.735 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x8000, CALLERID
>   param2=x0,
>   param3=x0,
> 3:50.15.745 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x10000, CALLEDID
>   param2=x0,
>   param3=x0,
> 3:50.15.745 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x4000, TRUNK
>   param2=x0,
>   param3=x0,
> 3:50.18.629 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x100, CONNECTED
>   param2=x10, <unknown flag(s)>
>   param3=x0,
```

```
> 3:50.18.629 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x8000, CALLERID
>   param2=x0,
>   param3=x0,
> 3:50.18.629 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x10000, CALLEDID
>   param2=x0,
>   param3=x0,
> 3:50.18.639 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x20000, CONNECTEDID
>   param2=x0,
>   param3=x0,
> 3:50.18.639 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x4000, TRUNK
>   param2=x0,
>   param3=x0,
> 3:50.19.460 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
>   param3=x0,
> 3:50.20.562 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
>   param3=x0,
> 3:50.21.663 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
```

```
>   param3=x0,
> 3:50.22.755 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
>   param3=x0,
> 3:50.23.866 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
>   param3=x0,
> 3:50.24.658 : received LINE_CALLINFO
>   device=x10322
>   cbInst=x0
>   param1=x1000000, CHARGINGINFO
>   param2=x0,
>   param3=x0,
> 3:50.24.688 : received LINE_CALLSTATE
>   device=x10322
>   cbInst=x0
>   param1=x1, IDLE
>   param2=x0,
>   param3=x0,
> 3:50.24.688 : lineGetCallStatus returned SUCCESS
> 3:50.25.859 : Calling lineGetCallInfo
>     hCall=x10322
>     lpCallInfo=x740048
> 3:50.25.869 : lineGetCallInfo returned SUCCESS
> LINECALLINFO
>   dwTotalSize=x10000
>   dwNeededSize=x1a8
>   dwUsedSize=x1a8
>   hLine=x10399
>   dwLineDeviceID=x13
>   dwBearerMode=x1, VOICE
>   dwMediaMode=x4, INTERACTIVEVOICE
>   dwCallID=xbb
>   dwRelatedCallID=xbb
>   dwCallParamFlags=x8e, IDLE BLOCKID ORIGOFFHOOK ONESTEPTRANSFER
```

```
>   dwCallStates=xfffb, IDLE OFFERING DIALTONE DIALING
>     RINGBACK BUSY SPECIALINFO CONNECTED PROCEEDING
>     ONHOLD CONFERENCED ONHOLDPENDCONF ONHOLDPENDTRANSFER
>     DISCONNECTED UNKNOWN
>   dwOrigin=x1, OUTBOUND
>   dwReason=x1, DIRECT
>   dwNumOwners=x1
>   dwTrunk=x1e79
>   dwCallerIDFlags=x8, ADDRESS
>   dwCallerIDSize=x4
>   dwCallerIDOffset=x164
>     00343031 xxxxxxxx xxxxxxxx xxxxxxxx  104.
>   dwCalledIDFlags=x8, ADDRESS
>   dwCalledIDSize=x8
>   dwCalledIDOffset=x154
>     33333333 00303031 xxxxxxxx xxxxxxxx  3333100.
>   dwConnectedIDFlags=x8, ADDRESS
>   dwConnectedIDSize=x8
>   dwConnectedIDOffset=x16c
>     33333333 00303031 xxxxxxxx xxxxxxxx  3333100.
>   dwRedirectionIDFlags=x20, UNKNOWN
>   dwRedirectionIDSize=x1
>   dwRedirectionIDOffset=x17c
>     xxxxxx00 xxxxxxxx xxxxxxxx xxxxxxxx  .
>   dwRedirectingIDFlags=x20, UNKNOWN
>   dwRedirectingIDSize=x1
>   dwRedirectingIDOffset=x17e
>     xxxxxx00 xxxxxxxx xxxxxxxx xxxxxxxx  .
>   dwAppNameSize=xd
>   dwAppNameOffset=x188
>     69706154 6f724220 72657377 xxxxxx00  Tapi Browser.
>   dwDisplayableAddressSize=x8
>   dwDisplayableAddressOffset=x144
>     33333333 00303031 xxxxxxxx xxxxxxxx  3333100.
>   dwChargingInfoSize=x4
>   dwChargingInfoOffset=x180
>     00343830 xxxxxxxx xxxxxxxx xxxxxxxx  084.  // 84 centimos
> 3:50.25.869 : Calling lineGetCallStatus
>     hCall=x10322
>     lpCallStatus=x740048
> 3:50.25.869 : lineGetCallStatus returned SUCCESS
```

```
> LINECALLSTATUS
>   dwTotalSize=x10000
>   dwNeededSize=x38
>   dwUsedSize=x38
>   dwCallState=x1, IDLE
>   dwCallPrivilege=x4, OWNER
>   dwCallFeatures=xa0000000, SETTREATMENT SETCALLDATA
>   tStateEntryTime[0]=x907d4
>   tStateEntryTime[1]=xd0001
>   tStateEntryTime[2]=x320001
>   tStateEntryTime[3]=x2b00018
> 3:50.35.2 : Calling lineClose
>     hLine=x10399
> 3:50.35.2 : lineClose returned SUCCESS
> 3:50.35.944 : Calling lineShutdown
>     hLineApp=x800003ee
> 3:50.35.954 : lineShutdown returned SUCCESS
```

Apéndice C.
Creación de la ayuda de la aplicación en formato CHM

El formato CHM o HTML comprimido, es el formato estandar de ayuda en aplicaciones de Windows. Está compuesto por una ventana dividida en dos vistas. La izquierda presenta los temas de la ayuda siendo posible hacer distintas búsquedas. La derecha muestra el fichero HTML de ayuda asociado al tema seleccionado en al vista izquierda.

Para la creación del fichero chm es necesario seguir 2 pasos:

Escribir los ficheros html de la ayuuda programa utilizando para ello cualquier editor html, pudiendo enlazar páginas entre si, incluir links externos, etc.

Con el programa **Html Help Workshop** distribuido gratuitamente por Microsoft y descargable desde http://msdn.microsoft.com/library/en-us/htmlhelp/html/hwMicrosoftHTMLHelpDownloads.asp agregar todos los ficheros html creados organizandolos jerárquicamente en la sección de "Contenidos (Contents)". El programa es muy fácil e intuitivo de usar y en la web de Microsoft hay mucha ayuda relacionada por lo que no se van a incluir un manual de cómo utilizar todas sus características. Cuando se termina de organizar los ficheros, el proyecto se compila y se genera el fichero .chm.

Referencias

[1] TAPI 2.2 Reference

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tapi/tapi2/tapi_2_2_reference.asp

[2] http://www.microsoft.com

[3] Novell TSAPI - http://developer.novell.com/support/sample/tids/doctsapi/doctsapi.htm

[4] Web oficial ECMA - http://www.ecma-international.org/

[5] Protocolo CSTA - http://www.ecma.ch/ecma1/TOPICS/TC32/TG11/CSTA.HTM

[6] Microsoft Platform SDK – http://www.microsoft.com/msdownload/platformsdk/sdkupdate/

[7] “Windows Telephony Programming, A Developer’s Guide to TAPI” Autor: Chris Sells

Ed: ADDISON-WESLEY – ISBN: 0-201-63450-3