

**TAPI – new age technology**  
**By Waseem Ahmad**  
**[wahmad@titledevelopments.com]**

**Abstract**

Today every business has to be client centric and must have up-to-date information for their customers when required. Call centers partly answer this business need by ensuring telephone calls are routed to agents who are able to answer a customer inquiry. Client Relationship Management [CRM] systems are the vehicle for providing call centre agents with the relevant customer information, thus helping to maintain a successful business relationship. Telephone call routing is the basic idea behind call centers, and TAPI is the backbone of this domain. Microsoft TAPI helps developers to write telephone applications with the features and functions required by modern businesses to fulfill their Call Centre and CRM requirements. This paper presents a brief introduction and history of TAPI and describes some of the work Title Developments has done in this area.

**Key words:**

ACD	Automatic Call Distribution
CLI	Customer Line Identifier
CRM	Client Relationship Management
CTI	Computer Telephone Integration
IVR	Interactive Voice Response
PBX	Private Branch Exchange
PSTN	Public Switched Telephone Network
QoS	Quality of Service
SPI	Service Provider Interface
TAPI	Telephony Application Programming Interface
TSPI	Telephony Service Provider Interface

**Introduction:**

A phone system for handling incoming calls can be enhanced by adding ACD capabilities. This typically is used by customer oriented businesses to handle

customer calls efficiently. The ACD system recognizes and answers the incoming calls according to instructions in a database, before sending the call to the most appropriate call centre agent or group of agents. Additional applications such as Blue Pumpkin's call reporting system, offer call centre management, information on the type and volume of calls and efficiency of the call centre agents.

TAPI is vital ingredient for telephony applications and ACD systems. It provides techniques for applications to easily support telephone communication, by facilitating [3]:

1. Direct connection to a telephone network.
2. Automatic phone dialing.
3. Data transmission and access.
4. Conference calling, voice mail, and caller ID.

**TAPI:**

The Telephony Application Programming Interface [TAPI] is an important API provided by Microsoft. TAPI is the set of APIs to manage and manipulate any type of communications link between a PC and the telephone line(s) [1]. The TAPI standard has a uniform set of functionality that can be used to communicate with any type of hardware that support TAPI-compliant service provider interface [SPI] [1]. Many different PC telephony models have been available for many years but the introduction of TAPI with its wide availability on the MS Windows platform has brought standardization and acceptance of one specific computer telephony interface.

Least Cost Routing is an example of an application which can use TAPI and gain benefits from TAPI standardization. This type of application looks at outbound calls and selects a call routing path via a

service provider who will provide the cheapest rate for making the call. The routing path chosen may vary depending on the call destination, time of day, and a service providers volume targets. TAPI may be used to look at the outbound call information and to reroute the call to the cheapest service provider once a decision has been made. TAPI standardization will allow the software vendor to support multiple telephony hardware vendors with the same software application.

The following advantages can be gained from using TAPI hardware and software [2]:

1. Automated configuration
2. Share the TAPI hardware with other TAPI compliant software.

According to Microsoft TAPI is applicable to [3]:

1. Basic voice calls on the Public Switched Telephone Network [PSTN]
2. Call center applications for tracking multiple agents
3. Modem control
4. PBX control
5. Interactive voice response [IVR] systems
6. Voice mail
7. Detailed phone device control

TAPI can be divided into two categories with respect to its devices. They are:

1. Line devices to model the physical telephony lines used to send and receive voice and data between locations [1].
2. Phone devices to model the desktop handset used to place and receive calls [1].

Each device has its own set of API calls.

### Lines:

In TAPI a physical line is represented as a line device object. A TAPI application can handle more than ONE line device at a

time. For example if an application is using Voice, Fax and Data lines, TAPI will consider these as three devices. The following figure shows the detail [1].

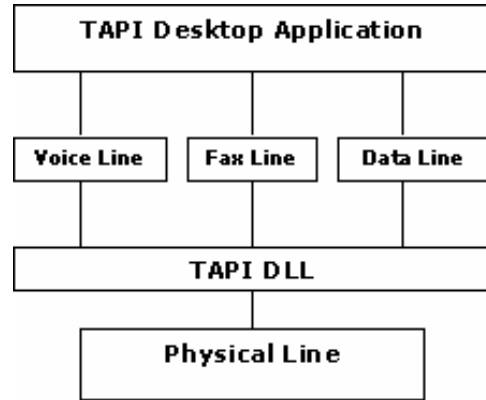


Fig. 1 TAPI with three devices [1]

### Phones:

The Phone Device is the second device modeled within TAPI. This model permits an Application programmer to create "Virtual Phone devices" within a PC application [1]. With a standard sound card, speakers and headphone, a simple PC can work as a phone set. There is no need for a 1 to 1 relationship between virtual phones devices and physical lines.

A more advanced example would be virtual phone devices created on a PBX. These devices could then be used by a PC server application to perform advanced call routing based on the callers CLI and a database lookup. The server application in this instance would support and control many virtual phone devices and perform routing on multiple calls at any given time.

### Hardware provider:

On the other side of the fence, hardware providers are required to support this new paradigm within their PBX's and other telephony equipment. Many hardware vendors including CISCO, NEC and Panasonic now support the TAPI standard

within their equipment; and this allows software application the ability to interact with their telephony hardware. Often the developer will find the majority of the functionality is consistent between hardware vendors implementation of TAPI. However, the developer should be aware that often the vendors use different name for the same functions.

### TAPI History

Microsoft TAPI is designed to enable and encourage programmers to develop hardware independent CTI solutions. TAPI's objective is to present vendors [software and hardware] with a consistent interface and device-independent programming model. This is a matter of defining a uniform interface and standard for further development in TAPI related applications. Thereby, the TAPI developer will be independent of the underlying hardware.

TAPI has evolved over a numbers of versions. *“Most of these versions involved changes to the TAPI and Telephony Service Provider Interface [TSPI] documentation sets; there are other implications for each version, namely, architectural differences, operating system variations, redistributables, and TSP development issues [3]”*. The following is a detail description of the different versions of TAPI [3].

TAPI version	Distribution
1.0 – 1.2	Beta versions that should not be used any longer.
1.4	Included in Windows 95
1.5	Included in Windows CE1.0.
2.0	Included in Windows NT® 4.0 SP3
2.1	Included in Windows 98 and Windows NT 4.0 SP4.
2.2 – 3x	Included in Windows Server 2003 family, Windows XP, and Windows 2000.

Table 1: Different versions of TAPI

### TAPI 1.4

TAPI 1.4 is the simplest and most basic form of TAPI available. It was introduced as a component of MS Windows 95 and based on a 16 bit architecture. However, it will support 32 bit applications without the developer being concerned about the limitations of 16 bit processing [3].

The TAPI and TSPI header files support the development of applications for both TAPI 1.4 and TAPI 2.x. There were no major changes to the API structure in these two versions. The only changes were in the API (specifically, Unicode support). For example:

```
#define          TAPI_CURRENT_VERSION
0x00010004
#include <tapi.h>
```

*“Note TAPI\_CURRENT\_VERSION should be defined for all TAPI applications [3]”*. In a TAPI 2.x development this is not necessary.

### TAPI 1.5

TAPI 1.5 was designed specifically for Windows® CE version 1.0. It is a subset of TAPI 1.4 and is only available on Windows CE, and is documented in the Windows CE developer documentation [3].

### TAPI 2.0

With TAPI 2.0, Microsoft delivered limited functional enhancement to the API. The main purpose of the release was to provide major architectural changes to TAPI which would allow it to work with the NT4.0 operating system and take advantage of its features - for instance full 32-bit support, services, and Unicode [3].

TAPI 2.0 also adds important features to the call center environment such as multiple call handling and better reporting; as well as support for Quality of Service [QoS].

## TAPI 2.1

Client/server functionality was the key feature of TAPI 2.1. *“With TAPI 2.1 and later, telephony hardware can be installed on a telephony server that can be accessed from any computer on a network [3].”* Administration DLLs were improved by providing a new set of functions to give access to remote resources.

## TAPI 2.x

With the popularity of TAPI and its functionality, new business applications have emerged. The Call Center is one of the major business areas which benefits from the improvements in TAPI functionality. Microsoft TAPI 2.2 offers the ability for telephony applications to provide better call center management functionality as more call information is available through the API.

## TAPI 3.x

With the development of COM-based technologies, TAPI version 3.x was enhanced to a COM-based API that combines classic and IP telephony functionality. Applications from simple voice calls over the Public Switched Telephone Network [PSTN] to multicast multimedia IP conferencing with quality of service [QoS] can be written using this version of TAPI.

There are four major components to TAPI 3.1:

1. COM API
2. TAPI Server
3. Telephony Service Providers [TSP]
4. Media Stream Providers [MSP]

TAPI 3.x provides the following enhancements to TAPI 2.1:

1. Support for IP telephony
2. Support for multiple-user conferencing
3. Support for NDIS 5.0-compliant devices

4. Uni-modem 5 support
5. Media stream control
6. Compatibility with the Common Object Model (COM)

## Our Experience:

For the last eight years Title Developments has been working actively in this domain and TAPI is now one of Title Developments areas of expertise. Over this period we have developed many applications and products for different client using TAPI.

Typically, we have developed products using the TAPI standard and use an architecture that with some minor changes can be used on different PBX platforms. This helps us to reduce our development time and therefore the costs for our clients.

This work has meant that we are now very experienced in advanced design techniques and the use of the latest tools and technologies. Clearly this is a major plus for Title Developments and its clients.

Because of the complex nature of these types of applications it is essential to have a very well equipped test lab which is able to simulate client's infrastructure. By having this kind of facility we can extensively test our product before delivery to our clients; and we can also reproduce scenarios reported to us.

As many of our clients also wish us to provide on-going support for their products, we include logging within our code. The text files generated contain details of all the activities performed by our applications. This helps Title Developments support staff, who can remotely access sites, to understand and quickly resolve any problems which may occur.

As one of our core business technologies, Title Developments have a state of art configuration management system. Internal and external release notes are

produced. Each and every version has extensive documentation for to allow further enhancements and changes.

**Our approach:**

TAPI based applications are not very simple to develop. Architecturally they are very complex. We have developed a generic TAPI based layer which communicates to the telephony hardware i.e. PBX. Our TAPI layer will handle all the TAPI based functionality and supports two main features of TAPI.

- a. Call Monitoring
- b. Call Controlling

The Telephony Server receives all the TAPI based events from the generic TAPI layer. The Telephony Server application uses these events to monitor for calls, process call information and apply business logic. Once a Telephony server has made a call routing decision it instructs the PBX to route the call to the appropriate destination whether this be a call centre agent or an IVR system.

Clients are connected to the Telephony Server through a TCP/IP channel. Microsoft SQL Server is used to record the system configuration and real time logging of call information and other call centre data such as agent availability.

The Telephony Server keeps the entire system configuration in memory to ensure speed of performance; a critical requirement for a Telephony application where the performance criteria for calls are measured in milliseconds.

Architecturally, our applications are developed as multi tier systems. Different tiers have different responsibilities like the TAPI layer, database layer, management layer, service layer, communication layer and so on. Multi threading is used throughout the application to optimize performance and sophisticated mechanisms have been employed to avoid deadlock and other system show stoppers.

Quality assurance, as always, is very important. To ensure effective quality assurance while testing telephony applications it is necessary have a fully equipped test lab with the latest network and PBX hardware. We have also found it necessary to develop in-house software which can be used to automate the testing of our telephony applications. Finally, due to the mission critical nature of telephony applications, the use of proven testing methodologies for functional, integration, stress and load testing is essential to ensure these applications are tested thoroughly.

**Conclusion:**

TAPI is now a sufficiently mature technology to be powerful enough to give software vendors the features and functionality necessary to produce telephony applications which give real benefits to their users. For businesses which use these applications the benefits will be seen in the additional quality of the service they are able to give to their customers.

**References:**

- [1]. <http://project.uet.itgo.com>
- [2]. <http://www.mtnsys.com>
- [3]. <http://msdn.microsoft.com>