iGuard™
Security System

API Software Development Kit Manual

Version 1.0

(for iGuard firmware version 2.6.7000a and above)

Copyright Notice and Proprietary Information
Copyright © 2000 Lucky Technology Co. All rights reserved. This software and documentation are owned by Lucky Technology Co., and furnished under a license agreement. The software and documentation may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Lucky Technology Co., or as expressly provided by the license agreement.

Right to Copy Documentation
The license agreement with Lucky Technology permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

Disclaimer
LUCKY TECHNOLOGY CO., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks
iGuard, the iGuard logo, and iGuard API Demo are trademarks of Lucky Technology Co. Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corp. All other brands or products may be trademarks, service marks or registered trademarks of their respective owners. All other products described in this book are trademarks of their respective holders and should be treated as such.

PRELIMINARY

iGuard™
Security System

Table of Contents

PRELIMINARY

PRELIMINARY

iGuard™
Security System

Preface

Welcome to the *iGuard Application Program Interface Software Development Kit (API SDK) V1.0.0.3 Manual*, a resource for developing personal, electronic identification products using Lucky Technology's proprietary software. With iGuard API, you can easily access and control the iGuard Terminal of Lucky Technology through the TCP/IP network.

About the API Software Development Kit

The iGuard API library includes the DLL file iGuard.dll, and the three header files iGuard.h, winapr.h and database.h. The functions are divided into four modules: Basic Module, Database Module, Server Module and RAS Module.

File Locations

Assuming the install drive is **C:**, the basic directory structure for SDK files is:

- **C:\iGuard**
- **C:\iGuard\bin**
- **C:\iGuard\doc**
- **C:\iGuard\examples**
- **C:\iGuard\include**
- **C:\iGuard\lib**

SDK System Requirements

To install the SDK, you need the following hardware and software:.

Table 1. System Requirements

Hardware:
- Pentium-Class PC or greater
- 16 MB RAM or greater
- 5 MB or greater free hard disk space.

Software:
- Microsoft Windows ® 95/98, NT, or 2000.
- TCP/IP Networking Support.
- Dial Up Networking (Windows ® 95/98) or RAS (Windows ® NT). ***
- Software development platform e.g. Microsoft Visual C++, Borland C++ Builder, Delphi.
- Winsock 2.0, (already built in with Windows ® 98, NT or 2000. Obtain update patch from Microsoft if using Windows ® 95.)

*** Needed only if RAS Function of iGuard API is used.

PRELIMINARY

Using This Manual

How to Quickly Find a Function in this Manual

Functions are grouped within the chapter by set. For example, `iGuard_Login,` and `iGuard_Logout` are presented together.

Conventions

The following typographical conventions are used:

- Files and applications appear as "file.ext".
- Functions appear as "`function ( )`".
- ◆ **Master** – Functions applied on iGuard Master Terminal only.
- ◆ **Connect** – Functions need a network connection between iGuard and local PC.
- ◆ **Login** – Functions need login before use (i.e. administrator's privilege), iGuard Terminal (Firmware Version 2.4 or later) has 3 level of password: Door Access, User Administration and System Administration. Different API function may needs different level of password authentication.

About iGuard Products and Services

General Information

For general information on our products and services, see the Lucky Technology home page on the World Wide Web at www.iguardsystem.com.

Technical Support

On-line support is available by e-mailing us at info@iguardsystem.com

Software Upgrade

For any software upgrade, please visit our web site or contact us by e-mail.

PRELIMINARY

Contact Information
Web Site : http://www.iguardsystem.com
E-Mail : info@iguardsystem.com

1. Basic Module

The Basic Module provides the standard routines for controlling and maintaining the iGuard device. Any applications that interact with the device must use the initialization and closing routines provided in this module.

USAGE NOTE: Call `iGuard_initialize()` to enable and initialize the API Library. To disable and shut down call `iGuard_Exit()` function.

PRELIMINARY

iGuard_Initialize()

To initializes iGuard API library.
Note: This function must be called prior to calling other functions.

Function Prototype

```
BOOL iGuard_Initialize(HWND hWnd, BOOL bShowMsg, BOOL
bShowLogin, DWORD dwWndMsg)
```

Inputs

hWnd	Window Handle of the Application Program.
bShowMsg	if 1 (TRUE), the program will show a popup error message box if error occurs.
bShowLogin	if 1 (TRUE), the program will show a popup login dialog box for user to login as the administrator when it is necessary in all the later functions that follow. If 0 (FLASE), the program will return error when it later calls the functions that require administrator's privilege.
DWORD dwWndMsg	Custom Message ID (e.g. WM_USER + 1) for iGuard API sending notification to application when necessary, like Server's Start / Stop.

Return Value

TURE if the DLL is successfully loaded and initialized.

Example

The following example initialize the iGuard API library with a custom message ID WM_IGUARDMSG (WM_USER + 1), and then shut down the API library.

```
#define WM_IGUARDMSG WM_USER + 1;

if(iGuard_Initialize(hWnd, TRUE, TRUE,  WM_IGUARDMSG)) {
     printf("iGuard API startup successfully.");
} else {
     printf("iGuard API startup failed.");
}

iGuard_Exit();
```

PRELIMINARY

```
iGuard_Exit()
```

To shut down and clean up the iGuard API library.
Note: This function must be called before your application shutdown (close), in order to let the iGuard API close and clean up the memory allocated and logout all the iGuard Terminals.

Function Prototype

```
CDECL_DLL(BOOL) iGuard_Exit(void);
```

Inputs

This function has no input parameter.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example initialize the iGuard API library with a custom message ID WM_IGUARDMSG (WM_USER + 1), and then shut down the API library.

```
#define WM_IGUARDMSG WM_USER + 1;

if(iGuard_Initialize(hWnd, TRUE, TRUE,  WM_IGUARDMSG)) {
     printf("iGuard API startup successfully.");
} else {
     printf("iGuard API startup failed.");
}

iGuard_Exit();
```

PRELIMINARY

```
iGuard_AboutDLL()
```

To shows a dialog box about the information of the iGuard API.

Function Prototype
```
CDECL_DLL(void) iGuard_AboutDLL();
```

Inputs
This function has no input parameter.

Return Value
This function does not return anything.

Example
The following show the about dialog box on screen.

```
iGuard_AboutDLL();
```

PRELIMINARY

`iGuard_DLLVersion()`

To get the version of the library currently running

Function Prototype
`CDECL_DLL(LPCTSTR) iGuard_DLLVersion()`

Inputs
This function has no input parameter.

Return Value
Pointer to ASCII string containing library version information

Example
The following example print out the version no. of iGuard API Library.

```
printf("iGuard API Version: %s", iGuard_DLLVersion());
```

PRELIMINARY

`ConvertToIP()`

To convert a string URL Address to IP Address without including the WINSOCK functions provided by Microsoft Windows Platform.

Function Prototype
`CDECL_DLL(DWORD) ConvertToIP(LPCSTR lpzUrl)`

Inputs
lpzURL pointer to the URL address
(e.g., 192.168.0.101 or www.device1.com)

Return Value
Converted IP Address (DWORD).

Example
This example shows how to convert a URL address (in string format) to IP address with iGuard API library.

```
char szURL[] = "http://www.iguardsystem.com";
DWORD dwIP = iGuard_DLLVersion(szURL);

printf("IP Address of %s is %d", szURL, dwIP);
```

PRELIMINARY

```
iGuard_CreateDefaultConfig()
```

This function creates default configuration in Windows' Registry.
Note: call this function will over write the original configuration.

Function Prototype
```
void iGuard_CreateDefaultConfig(void)
```

Inputs
This function has no input parameter.

Return Value
This function does not return anything.

Example
This example shows how to create a default setting of iGuard API Library in Microsoft Windows' registry.

```
iGuard_CreateDefaultConfig();
```

The function create a folder in registry (may vary with different iGuard API Version):
- HKEY_LOCAL_MACHINE\SOFTWARE\Lucky Technology Co.\iGuard\Config

and the following key value (may vary with different iGuard API Version):
- Host Description: iGuard API Server
- Service Port: (0x00000c10) 3088

PRELIMINARY

`iGuard_SaveConfig()`

To save the Response Server 's configuration.

Function Prototype
`void iGuard_SaveConfig(void)`

Inputs
This function has no input parameter.

Return Value
This function does not return anything.

PRELIMINARY

```
iGuard_GetVersion()
```

◆ Connect

To get the iGuard Terminal Firmware version information.

Function Prototype

```
BOOL iGuard_GetVersion(LPCTSTR lpzURL, LPTSTR lpzBuffer,
int nMaxLen)
```

Inputs

lpzURL	pointer to the URL address of the iGuard Terminal. (e.g., 192.168.0.101 or www.device1.com)
lpzBuffer	pointer to the buffer to hold the version no.
nMaxLen	the buffer length

Return Value

TURE if the function is successful. FALSE if the function fails (such as buffer overflow, device not accessible ... etc.)

Example:

The following example shows how to get the firmware version from iGuard.

```
char lpzURL[] = "192.168.0.100";
char lpzBffer[256];

if(iGuard_GetVersion(lpzURL, lpzBuffer, 256)) {
     printf("The version of iGuard with IP Address %s :
%s", lpzURL, lpzBuffer);
}
```

PRELIMINARY

```
iGuard_GetTerminalName()
```

◆ Connect

To get the iGuard Terminal Name.

Function Prototype

```
BOOL iGuard_GetTerminalName(LPCTSTR lpzURL, LPTSTR
lpzBuffer, int nMaxLen)
```

Inputs

lpzURL	pointer to the URL address of the iGuard Terminal. (e.g., 192.168.0.101 or www.device1.com)
lpzBuffer	pointer to the buffer to hold the terminal name.
nMaxLen	the buffer length

Return Value

TURE if the function is successful. FALSE if the function fails (such as buffer overflow, device not accessible ... etc.)

Example:

The following example shows how to get the terminal name from iGuard.

```
char lpzURL[] = "192.168.0.100";
char lpzBuffer[256];

if(iGuard_GetTerminalName(lpzURL, lpzBuffer, 256)) {
     printf("The terminal name of iGuard with IP Address %s
: %s", lpzURL, lpzBuffer);
}
```

PRELIMINARY

```
iGuard_Unlock()
```

◆ Connect ◆ Login

To unlock the iGuard Terminal's Door. This function requires privilege of door access, user or administrator.

Function Prototype

```
BOOL iGuard_Unlock(LPCTSTR lpzUrl)
```

Inputs

lpzUrl pointer to the URL address of the iGuard Terminal.

Return Value

TURE if the function is successful. FALSE if the function.

Example:

The following example shows how to signal an iGuard to unlock.

```
char lpzURL[] = "192.168.0.100";

if(iGuard_Login(lpzURL, “Admin”, “123”) ) {
     if(iGuard_Unlock(lpzURL)) {
          printf("The iGuard w/ IP %s is now unlocked”
               , lpzURL);
     }
}
```

PRELIMINARY

iGuard_SetDateTime()

◆ Connect ◆ Login

To set the system time of the iGuard Terminal. This function requires administrator's privilege.
Note: if change the date / time of iGuard Master Terminal, all the slave iGuard Terminal's date / time will be synchronized will master automatically.

Function Prototype

```
BOOL iGuard_SetDateTime(LPCTSTR lpzUrl, PSYSTEMTIME lpTime)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal.
lpTime	pointer to the system structure PSYSTEMTIME

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The following example shows how to synchronize the clock of iGuard with your PC's Clock.

```
char lpzURL[] = "192.168.0.100";
SYSTEMTIME sysTime;

GetSystemTime(&sysTime);

if(iGuard_Login(lpzURL, "Admin", "123") ) {
     if(iGuard_SetDateTime(lpzURL, &sysTime)) {
          printf("The clock of iGuard with IP Address %s
and your PC are now synchronized", lpzURL);
     }
}
```

PRELIMINARY

```
iGuard_Reboot()
```

◆ Connect ◆ Login

To reboot the iGuard Terminal. This function requires administrator's privilege.

Function Prototype

```
BOOL iGuard_Reboot(LPCTSTR lpzUrl)
```

Inputs

lpzUrl pointer to the URL address of the iGuard Terminal.

Return Value

TURE if the function is successful. FALSE if the function fails.

```
if(iGuard_Login(lpzURL, "Admin", "123") ) {
     iGuard_Reboot(lpzURL);
}
```

PRELIMINARY

`iGuard_Login()`

◆ Connect ◆ Login

To login the iGuard Terminal. This function will enable all the administrator's privilege commands.

Function Prototype

```
BOOL iGuard_Login(LPCTSTR lpzUrl, LPCTSTR lpzUserName,
LPCTSTR lpzPassword)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal.
lpzUserName	pointer to the login user name of Administrator.
lpzPassword	pointer to the login password of Administrator.

Return Value

TURE if the function is successful. FALSE if the function fails.

Remarks:

This command only setup the login information in the library which will be used when necessary (i.e. the functions that need administrator's privilege). Application (or user) only need to login the iGuard terminal once. After login successfully, the login information will be sent automatically if necessary. However, these login information will be cleared by calling `iGuard_Logout()` function or shutting down the application (i.e. shut down the iGuard API library).

Important: The iGuard API library WILL NOT store any login information. Once the API is shut down, everything will be gone. It is the duty of the application to decide whether keeping the login information or not.

Example:

The following example shows how to login an iGuard terminal.

```
char lpzURL[] = "192.168.0.100";

if(iGuard_Login(lpzURL, “Admin”, “123”) ) {
     if(iGuard_Unlock(lpzURL)) {
          printf("The iGuard w/ IP %s is now unlocked”
               , lpzURL);
     }
}
```

PRELIMINARY

```
iGuard_Logout()
```

To logout all iGuard Terminal and clear all the login information. This function will disable all the administrator's privilege commands.

Function Prototype
```
void iGuard_Logout(void)
```

Inputs
This function has no input parameter.

Return Value
This function does not return anything.

Example:
The following example shows how to logout all iGuard terminal.

```
iGuard_Logout();
```

PRELIMINARY

`iGuard_LoginDialogBox()`

◆ Connect ◆ Login

To show the default login dialog box for iGuard Terminal Login. The login information will be kept in iGuard API library after successful login, and will be cleared by calling function `iGuard_Logout()` or shutting down the iGuard API Library.

Function prototype

```
CDECL_DLL(BOOL) iGuard_LoginDialogBox(LPCTSTR lpzUrl)
```

Inputs

lpzUrl pointer to the URL address of the iGuard Terminal.

Return Value

TRUE if fucntion login iGuard terminal successful.

Example

This example shows how to use the default login dialog box to login the iGuard Terminal.

```
char lpzURL[] = "192.168.0.100";

if(iGuard_LoginDialogBox(lpzURL)) {
     printf("Successful login iGuard Terminal at IP %s",
          lpzURL);
}
```

PRELIMINARY

```
iGuard_GetMasterIP()
```

◆ Connect

To get the IP address of the iGuard Master Terminal from any iGuard Terminal.
Note: some of iGuard API function only valid when apply to iGuard Master Terminal, e.g. All the functions in Database Module.

Function Prototype

```
DWORD iGuard_GetMasterIP(LPCTSTR lpzUrl)
```

Inputs

lpzUrl pointer to the URL address of the iGuard Terminal.

Return Value

The IP address of the Master Terminal.

Example:

The following example shows how to retrieve the IP Address of master unit from one of the iGuard terminal in your network.

```
char lpzURL[] = "192.168.0.102";
DWORD dwIP = iGuard_GetMasterIP(lpzURL);
```

PRELIMINARY

`iGuard_IsMaster()`

◆ Connect

To determine if the iGuard Terminal is configured as Master unit or Slave unit. Application can also use the function `iGuard_GetMasterIP()` to retrieve the Master Unit's IP address.

Function Prototype

```
BOOL iGuard_IsMaster(LPCTSTR lpzUrl)
```

Inputs

lpzUrl pointer to the URL address of the iGuard Terminal.

Return Value

TRUE if the iGuard Terminal is configured as Master unit. FALSE if the iGuard Terminal is configured as Slave unit.

Example

The following example shows how to determine the iGuard Terminal is configured as Master or Slave.

```
char lpzURL[] = "192.168.0.102";

if(iGuard_IsMaster(lpzURL)) {
     printf("iGuard at IP Address %s is a Master", lpzURL);

}
```

PRELIMINARY

```
iGuard_GetSerialNo()
```

◆ Connect

To get the serial number of the iGuard Terminal. This is a unique serial number.

Function Prototype

```
BOOL iGuard_GetSerialNo(LPCTSTR lpzURL, LPTSTR lpzBuffer,
int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal.
lpzBuffer	pointer to the buffer to hold the version no.
nMaxLen	the buffer length

Return Value

TURE if the function is successful. FALSE if the function fails (such as buffer overflow, device not accessible ... etc.)

Example

The following example shows how to retrieve the Serial No. of an iGuard Terminal.

```
char lpzURL[] = "192.168.0.102";
char lpzBuffer[40];

if(iGuard_GetSerialNo(lpzURL, lpzBuffer, 40)) {
     printf("S/N of iGuard at IP Address %s is %s",
          lpzURL, lpzBuffer);
}
```

PRELIMINARY

iGuard_GetLastError()

To get the error code of the last error occurred.

Function Prototype
```
int iGuard_GetLastError(void)
```

Return Value
The error code of the last error occurred.

Example
The following example shows how to get the last error message from iGuard API Library.

```
int nErrorCode = iGuard_GetLastError();
printf('Last error is : %s (%d)",
     iGuard_GetErrorMessage(nErrorCode), nErrorCode);
```

PRELIMINARY

`iGuard_GetErrorMessage()`

To get the corresponding error message of the error code.

Function Prototype
```
LPCTSTR iGuard_GetErrorMessage(int nErrorCode)
```

Inputs
nErrorCode the error code

Return Value
Pointer to the printable ASCII string containing the corresponding error message of the error code.

Example:
The following example shows how to retrieve an error code and message when library function failed.

```
int nErrCode = iGuard_GetLastError();
printf("Last error is:%s",iGuard_GetErrorMessage(nErrCode);
```

PRELIMINARY

iGuard_SetProgressCallback()

Set up a call back function for iGuard API when sending or receiving something big to / from iGuard Terminal. iGuard API will call your program to report the progress of transmission.

Function Prototype

```
void iGuard_SetProgressCallback(ProgressCallbackFunc
*pCallback)
```

Inputs

pCallback pointer to the call back function.

Return Value

This function does not return anything.

Note

Function that will receive the progress information should be declarated as a CALLBACK type function. For example: static void __stdcall ShowProgress(int nPercentage);

Example:

The follow example shows how to setup a call back function to display transmission status.

```
iGuard_SetProgressCallback(ShowProgressProc);
:::
//Your Program Code
:::
static void __stdcall ShowProgressProc(int nPercentage)
{
     printf("Transmission : %d %%", nPercentage);
}
```

PRELIMINARY

```
iGuard_ShowServerIcon()
```

Show or Hide the iGuard Response Server's Tray Icon.

Function Prototype
```
BOOL iGuard_ShowServerIcon(BOOL bShow)
```

Inputs
bShow TRUE to Show the iGuard Response Server Tray Icon.

Return Value
TURE if the function is successful. FALSE if the function fails.

Example:
The follow example shows how to show the iGuard Response Server's Tray Icon.

```
if(iGuard_ShowServerIcon(TRUE)) {
     printf("The server's icon shown on the Tray");
}
```

PRELIMINARY

2. Database Module

The Database Module provides the routines to access the database inside the iGuard Terminal. Please note that you can access the databases inside an iGuard Master Terminal only. Changes in database inside the Master Terminal will be replicated to it's Slave Terminal automatically.

Databases are defined by Name. Here are the databases' name that you can access in an iGuard Terminal (Master Unit only).

Table 2. iGuard Databases

Short Name	Full Name	Notes
EMPY	Employee	Employee's Information
TMLG	Access Log	Add / Delete only with manually added record
DEPT	Department	Department's Information
SLAV	Terminals	Slave Terminal List, Read Only
EMPF	Employee Full Record	Employee's Information with fingerprints' template included.

(Short Name is used when using the functions in the database module.)

For details information about the definition of database and it's record formation, please refer to the header file "Database.h".

Important:
The field called RCDID (Record ID, the record's Reference No.) in each record that returned by database function, should be REMAIN UNCHANGED. Changing this field's value may result in database corruption.

PRELIMINARY

```
iGuard_DatabaseTerminalList()
```

◆ Connect ◆ Master

Get the list of available iGuard Terminals.

Function Prototype

```
BOOL iGuard_DatabaseTerminalList(LPCSTR lpzUrl, LPTSTR
lpzTerminalLIst, int nMaxLen, int *nListLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzTerminalList	pointer to the buffer that will receive the terminal list. (all terminal ID are separated by a character '|')
nMaxLen	length of the buffer in bytes.
nListLen	pointer to integer that will receive the length of the terminal list, in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Note

If use this function on an iGuard Terminal configured as Slave unit, the iGuard Terminal will only return a terminal list that contain only the terminal itself.

Example:

The follow example shows how to retrieve the terminal list from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
char lpzBuffer[256];
int nMaxLen = 256;
int nListLen = 0;

if(iGuard_DatabaseTerminalList(lpzURL, lpzBuffer, nMaxLen,
&nListLen)) {
     printf("The terminals in your network are : %s");
}
```

PRELIMINARY

```
iGuard_DatabaseDepartmentList()
```

◆ Connect ◆ Master

Get the list of available Departments (Only the Departments' ID field will be returned in the list).

Function Prototype

```
BOOL iGuard_DatabaseDepartmentList(LPCTSTR lpzUrl, LPTSTR
lpzTerminalLIst, int nMaxLen, int *nListLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDepartmentList	pointer to the buffer that will receive the department list.
nMaxLen	length of the buffer in bytes.
nListLen	pointer to integer that will receive the length of the terminal list, in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the department list from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
char lpzBuffer[256];
int nMaxLen = 256;
int nListLen = 0;

if(iGuard_DatabaseDepartmentList(lpzURL, lpzBuffer,
nMaxLen, &nListLen)) {
     printf("The departments defined in iGuard: %s");
}
```

PRELIMINARY

```
iGuard_DatabaseEmployeeList()
```

◆ Connect ◆ Master

Get the list of Employees (Only Employees' ID will be returned in the list).

Function Prototype

```
BOOL iGuard_DatabaseEmployeeList(LPCTSTR lpzUrl, LPTSTR
lpzTerminalLIst, int nMaxLen, int *nListLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzEmployeeList	pointer to the buffer that will receive the employee list.
nMaxLen	length of the buffer in bytes.
nListLen	pointer to integer that will receive the length of the terminal list, in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the employee list from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
char lpzBuffer[256];
int nMaxLen = 256;
int nListLen = 0;

if(iGuard_DatabaseEmployeeList(lpzURL, lpzBuffer, nMaxLen,
&nListLen)) {
     printf("The Employees defined in iGuard: %s");
}
```

PRELIMINARY

```
iGuard_DatabaseFirst()
```

◆ Connect ◆ Master

Get the first record from a database.

Function Prototype

```
BOOL iGuard_DatabaseFirst(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the buffer of database record.
nMaxLen	length of the buffer in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the first employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

if(iGuard_DatabaseFirst(lpzURL, “EMPY”, &dbEmployeeRcd,
sizeof(DB_EMPLOYEE))) {
     printf("Name of first employee defined in iGuard: %s,
%s", dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabasePrevious()
```

◆ Connect ◆ Master

Get the previous record from a database.

Function Prototype

```
BOOL iGuard_DatabasePrevious(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the buffer of database record (it must contain the current record).
nMaxLen	length of the buffer in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the previous employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

::::
//Do some record navigating
//Note : Don't change value of Field rcdid.
::::

if(iGuard_DatabasePrevious(lpzURL, "EMPY", &dbEmployeeRcd,
sizeof(DB_EMPLOYEE)) {
     printf("Name of employee previous record in iGuard:
%s, %s", dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseNext()
```

◆ Connect ◆ Master

Get the next record from a database.

Function Prototype

```
BOOL iGuard_DatabaseFirst(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the buffer of database record (it must contain the current record).
nMaxLen	length of the buffer in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the next employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

::::
//Do some record navigating
//Note : Don't change value of Field rcdid.
::::

if(iGuard_DatabaseNext(lpzURL, "EMPY", &dbEmployeeRcd,
sizeof(DB_EMPLOYEE)) {
     printf("Name of employee previous record in iGuard:
%s, %s", dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseLast()
```

◆ Connect ◆ Master

Get the last record from a database.

Function Prototype

```
BOOL iGuard_DatabaseFirst(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the buffer of database record.
nMaxLen	length of the buffer in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the last employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

if(iGuard_DatabaseLast(lpzURL, "EMPY", &dbEmployeeRcd,
sizeof(DB_EMPLOYEE))) {
     printf("Name of last employee defined in iGuard: %s,
%s", dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseFind()
```

◆ Connect ◆ Master

Find a record in database.

Function Prototype

```
BOOL iGuard_DatabaseFind(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPCTSTR lpzID, LPTSTR pRecord, int
nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
lpzID	pointer to the ID to be found. For example, Employee ID.
pRecord	pointer to the buffer of database record.
nMaxLen	length of the buffer in bytes.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the find an employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;
Char lpzEmployeeID[] = "A0001";

if(iGuard_DatabaseFind(lpzURL, "EMPY", lpzEmployeeID,
&dbEmployeeRcd, sizeof(DB_EMPLOYEE))) {
     printf("Name of employee %s in iGuard: %s, %s",
          lpzEmployeeID, dbEmployeeRcd.lastname,
               dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseDelete()
```

◆ Connect ◆ Master ◆ Login

Delete a record in database. This function requires administrator's privilege.

Function Prototype

```
BOOL iGuard_DatabaseDelete(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPCTSTR lpzID)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
lpzID	pointer to the ID to be deleted. For example, Employee ID.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to delete an employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
Char lpzEmployeeID[] = “A0001”;

if(iGuard_DatabaseDelete(lpzURL, “EMPY”, lpzEmployeeID)) {
     printf("Employee %s delete from iGuard ",
          lpzEmployeeID);
}
```

PRELIMINARY

```
iGuard_DatabaseDeleteAccessLog()
```

◆ Connect ◆ Master ◆ Login

Delete a manually added access log record in database. This function requires administrator's privilege.

Function Prototype

```
BOOL iGuard_DatabaseDeleteAccessLog(LPCTSTR lpzUrl, LPCTSTR
lpzEmployeeID, const FILETIME *lpLogTime)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzEmployeeID	pointer to the ID of Employee.
lpLogTime	pointer to a FILETIME structure, contains date and time of access log record to be deleted.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to delete a manually added access log record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
Char lpzEmployeeID[] = “A0001”;
FILETIME logTime;
SYSTEMTIME sysTime;

sysTime.wYear = 2000;
sysTime.wMonth = 1;
sysTime.wDay = 10;
sysTime.wHour = 10;
sysTime.wMinute = 0;
sysTime.wSecond = 0;
sysTime.wMilliseconds = 0;
SystemTimeToFileTime(&sysTime, &logTime);

if(iGuard_DatabaseDeleteAccessLog(lpzURL, lpzEmployeeID,
&logTime)) {
     printf("Employee’s access log delete from iGuard ");
}
```

PRELIMINARY

```
iGuard_DatabaseInsert()
```

◆ Connect ◆ Master ◆ Login

Insert a new record to database. This function requires administrator's privilege.

Function Prototype

```
BOOL iGuard_DatabaseInsert(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the record to be inserted.
nMaxLen	length of record buffer.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to add an employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

memset(&dbEmployeeRcd, 0, sizeof(dbEmployeeRcd));
strcpy(dbEmployeeRcd.id, "A0001");
strcpy(dbEmployeeRcd.lastname, "Lee");
strcpy(dbEmployeeRcd.firstname, "Tom");
strcpy(dbEmployeeRcd.othername, "Manager");

if(iGuard_DatabaseInsert(lpzURL, "EMPY", &dbEmployeeRcd,
sizeof(DB_EMPLOYEE))) {
     printf("Employee %s, %s added in iGuard",
          dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseUpdate()
```

◆ Connect ◆ Master ◆ Login

Update a record in database. This function requires administrator's privilege.

Function Prototype

```
BOOL iGuard_DatabaseUpdate(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPTSTR pRecord, int nMaxLen, int nOption)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
pRecord	pointer to the record to be inserted.
nMaxLen	length of record buffer.
nOption	Record's addition information, (Currently there's only one option, DB_UPDOPT_SAVEPASSWORD, used in Employee Record, if this bit is set, the password field in the record is saved. If clear, put 0 in this parameter, the password field is ignored.)

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to add an employee record from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
DB_EMPLOYEE dbEmployeeRcd;

memset(&dbEmployeeRcd, 0, sizeof(dbEmployeeRcd));
strcpy(dbEmployeeRcd.id, "A0001");
strcpy(dbEmployeeRcd.lastname, "Lee");
strcpy(dbEmployeeRcd.firstname, "Tom");
strcpy(dbEmployeeRcd.othername, "Manager");
strcpy(dbEmployeeRcd.password, "1234");

if(iGuard_DatabaseInsert(lpzURL, "EMPY", &dbEmployeeRcd,
sizeof(DB_EMPLOYEE), DB_UPDOPT_SAVEPASSWORD)) {
     printf("Employee %s, %s's record is updated",
          dbEmployeeRcd.lastname, dbEmployeeRcd.firstname);
}
```

PRELIMINARY

```
iGuard_DatabaseRecordCount()
```

◆ Connect ◆ Master

Count the total number of records in a database.

Function Prototype

```
int iGuard_DatabaseRecordCount(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.

Return Value

Total number of records in a database if function successfully, 0 if database empty or function fails.

Example:

The follow example shows how to add an employee record from an iGuard Master.

```
printf("There're %d records in Employee Database",
     iGuard_DatabaseInsert(lpzURL, "EMPY");
```

PRELIMINARY

```
iGuard_DatabaseGetOption()
```

◆ Connect ◆ Master

Get the option field of a record. For example, departments of a employee and terminals of a department.
Note: data is separated by character '|'

Function Prototype

```
BOOL iGuard_DatabaseGetOption(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPCTSTR lpzID, LPTSTR pOption, int
nMaxLen)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
lpzID	pointer to the record's ID, for example Employee ID.
pOption	pointer to a buffer for holding the options.
nMaxLen	length of buffer.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to retrieve the option field (departments) of an employee from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
char lpzBuffer[100];
int nMaxLen = 100;

memset(&lpzBuffer, 0, nMaxLen);

if(iGuard_DatabaseGetOption(lpzURL, "EMPY", "A0001",
lpzBuffer, nMaxLen)) {
     printf("Employee A0001 belong to %s", lpzBuffer);
}
```

PRELIMINARY

```
iGuard_DatabaseSaveOptions()
```

◆ Connect ◆ Master ◆ Login

Save the option field to database.

Function Prototype

```
BOOL iGuard_DatabaseSaveOptions(LPCTSTR lpzUrl, LPCTSTR
lpzDatabaseName, LPCTSTR lpzID, LPCTSTR pOption)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDatabaseName	pointer to the database name.
lpzID	pointer to the record's ID.
pOption	pointer to the options.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to add the option field (departments) of an employee from an iGuard Master.

```
char lpzURL[] = "192.168.0.100";
char lpzDepartment = "EVERYONE|MARKETING";
int nMaxLen = 100;

if(iGuard_DatabaseSaveOption(lpzURL, "EMPY", "A0001",
lpzDepartment)) {
     printf("Employee A0001 belong to %s", lpzDepartment);
}
```

PRELIMINARY

```
iGuard_DatabaseCheckRestriction()
```

◆ Connect ◆ Master

Check the time restriction of a department.

Function Prototype

```
BOOL iGuard_DatabaseCheckRestriction(void
*pDepartmentRecord, int nLen, int nDay, int nHour, int
nMinute)
```

Inputs

pDepartmentRecord	pointer to a department record.
nLen	length of the department record (for version checking purpose).
nDay	week day, 0 for Sunday, 6 for Saturday and 7 for Holiday.
nHour	hour, 0 - 23 (in 24 Hours format).
nMinute	minute, 0 - 59.

Return Value

TURE if the access for specified time is enabled. FALSE if disabled.

Example:

The follow example shows how to check the restriction of a department at Monday, 12:05 AM.

```
char lpzURL[] = "192.168.0.100";
DB_DEPARTMENT dbDeptRcd;
int nMaxLen = 100;

if(iGuard_DatabaseFind(lpzURL, “DEPT”, “EVERONE”,
&dbDeptRcd, sizeof(DB_DEPARTMENT))) {
     if(iGuard_DatabaseCheckRestriction(&dbDeptRcd,
          sizeof(DB_DEPARTMENT), 1, 12, 5)) {
          printf(“Employee of EVERYONE have right”);
     } else {
          printf(“Employee of EVERYONE don’t have right”);
     }
}
```

PRELIMINARY

```
iGuard_DatabaseSetRestriction()
```

◆ Connect ◆ Master ◆ Login

Set the time restriction of a department.

Function Prototype

```
BOOL iGuard_DatabaseSetRestriction(void *pDepartmentRecord,
int nLen, int nDay, int nHour, int nMinute, BOOL bEnable)
```

Inputs

pDepartmentRecord	pointer to a department record.
nLen	length of the department record (for version checking purpose).
nDay	week day, 0 for Sunday, 6 for Saturday and 7 for Holiday.
nHour	hour, 0 - 23 (in 24 Hours format).
nMinute	minute, 0 - 59.
bEnable	enable / disable access.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example:

The follow example shows how to set the restriction of a department at Monday, 12:05 AM to TRUE.

```
char lpzURL[] = "192.168.0.100";
DB_DEPARTMENT dbDeptRcd;
int nMaxLen = 100;

if(iGuard_DatabaseFind(lpzURL, “DEPT”, “EVERONE”,
&dbDeptRcd, sizeof(DB_DEPARTMENT))) {
     if(iGuard_DatabaseSetRestriction(&dbDeptRcd,
          sizeof(DB_DEPARTMENT), 1, 12, 5, TRUE)) {
          printf(“EVERYONE restriction set ok”);
     }
}
```

PRELIMINARY

`iGuard_DatabaseAddHoliday()`

◆ Connect ◆ Master ◆ Login

Add a company holiday to the database.

Function Prototype

```
BOOL iGuard_DatabaseAddHoliday(LPCTSTR lpzUrl, LPCTSTR
lpzDate)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDate	pointer to a date to be added (in MM/DD/YYYY format).

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to add a holiday to iGuard database.

```
char lpzURL[] = "192.168.0.100";
char lpzDate[] = "1/1/2000";
if(iGuard_DatabaseAddHoliday(lpzURL, lpzDate)) {
     printf("Date %s set as holiday", lpzDate);
}
```

PRELIMINARY

```
iGuard_DatabaseRemoveHoliday()
```

◆ Connect ◆ Master ◆ Login

Remove a company holiday to the database.

Function Prototype

```
BOOL iGuard_DatabaseRemoveHoliday(LPCTSTR lpzUrl, LPCTSTR
lpzDate)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzDate	pointer to a date to be Remove (in MM/DD/YYYY format).

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to remove a holiday to iGuard database.

```
char lpzURL[] = "192.168.0.100";
char lpzDate[] = "1/1/2000";
if(iGuard_DatabaseRemoveHoliday(lpzURL, lpzDate)) {
     printf("Date %s removed from holiday list", lpzDate);
}
```

PRELIMINARY

iGuard_DatabaseAddIOTrigger()

◆ Connect ◆ Master ◆ Login

Add an In / Out Trigger time to the database.

Function Prototype

```
BOOL iGuard_DatabaseAddIOTrigger(LPCTSTR lpzUrl, LPCTSTR
lpzTime, BOOL bIn)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzTime	pointer to a time to be added (in HH:MM, 24 Hour format).
bIn	TRUE for In, FALSE for Out.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to add a IO Trigger time to iGuard database.

```
char lpzURL[] = "192.168.0.100";
char lpzTime[] = "7:30";
if(iGuard_DatabaseAddIOTrigger(lpzURL, lpzTime, TRUE)) {
     printf("Terminal will change to IN at %s", lpzTime);
}
```

PRELIMINARY

```
iGuard_DatabaseRemoveIOTrigger()
```

◆ Connect ◆ Master ◆ Login

Remove an In / Out Trigger time to the database.

Function Prototype

```
BOOL iGuard_DatabaseRemoveIOTrigger(LPCTSTR lpzUrl, LPCTSTR
lpzTime)
```

Inputs

lpzUrl	pointer to the URL address of the iGuard Terminal. Please note that this terminal must be configured as Master unit.
lpzTime	pointer to a time to be added (in HH:MM, 24 Hour format).

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to remove an IO Trigger Time to iGuard database.

```
char lpzURL[] = "192.168.0.100";
char lpzTime[] = “7:30”;
if(iGuard_DatabaseRemoveIOTrigger(lpzURL, lpzTime)) {
     printf(“IO Trigger time %s removed.”, lpzTime);
}
```

PRELIMINARY

3. Server Module

Server module create an iGuard Response Server to receive the real time access log send from iGuard terminals.

Notes:

- The Response Server is a software module that runs in the background of the PC. It is indicated by an Icon at the system tray (software configurable). It notifies the Master iGuard unit the present of this server, and the Master unit will automatically update the new access log records to the PC.
- A TCP/IP's Port need to be assigned to the iGuard Response Server (Default Port No. is 3080), if the port is already assigned to some other application like Web Server. Application can assign another port for iGuard Response Server by calling the `iGuard_SetServerPort()` function, and then call the `iGuard_SaveConfig()` function to save the new setting to registry.
- Application using Response Server need to register the local PC's Response Server to the iGuard Master Terminal by calling the function `iGuard_RegisterServer()`. If there's more that one iGuard Master in your network, your application need to register each iGuard Master (unlimited, depends on your system resource). After register, new access log will automatically send to the local PC as long as the PC and iGuard Master Terminal are online.
- Old access log can be retrieved by using functions available in Database Module. (Please refer to Database Module section for details).
- For Security Reason, login information (Administration Level) is necessary when register to an iGuard Master Terminal (i.e. using function `iGuard_Login()`).
- If iGuard_UnRegisterServer is called, new access log record will not send to Response Server.

Important

Application need to call this function to Register iGuard Master terminal in order to receive access log automatically. However, there's some limitation of the iGuard API Response Server.

1. The access log that before your application register by using function iGuard_RegisterServer will not send to the Response Server. You need to import it by the API's database function if necessary.

2. iGuard Master will stop sending the access log to Response Server if it's failed once (i.e. network connection failed or application is closed).

3. Once Response Server registered, it will automatically re-register the Response Server every 2 minutes, for example, if the network is downed, iGuard Master will stop sending (i.e. item 2 above), the access log will send to Response Server after network connection restored and it automatically re-register.

4. Application need to register the Response Server to the iGuard Master terminal whenever it's start, otherwise, access log will not sent to Response Server (i.e. condition of Item 2).

5. **Don't call the iGuard_UnRegisterServer** to un-register the Response Server from iGuard Master, this function is for maintenance purpose only (i.e. Response Server is no longer necessary). Your application can shutdown any time without telling the iGuard Master.

PRELIMINARY

`iGuard_IsServerStarted()`

Check if Response Server started.

Function prototype
`CDECL_DLL(BOOL) iGuard_IsServerStarted(void)`

Inputs
This function does not take parameter.

Return Value
TRUE if Response Server started, FALSE otherwise.

Example
The following example shows how to whether the Response Server is started.

```
if(iGuard_IsServerStarted()) {
     printf("iGuard Response Server Started");
}
```

PRELIMINARY

`iGuard_StartServer()`

Start a Response Server in local PC to receive the data sent from iGuard terminal.

Function prototype

```
CDECL_DLL(BOOL) iGuard_StartServer(RecvAccessLogFunc
*pCallback)
```

Inputs

pCallback — pointer to callback function in application to receive data when API received data from iGuard terminal. If set this parameter to NULL, all the access logs received will be saved to a file called 'access.log' in current directory.

Return Value

TRUE if Response Server started successful.

Note: The Response Server is a software module that runs in the background of the PC. It is indicated by an Icon at the system tray. It notifies the Master iGuard unit the present of this server, and the Master unit will automatically update the new access log records to the PC. Please refer to the beginning of the document for more details about the Response Server.

Example

The following example shows how to start an iGuard Response Server in local PC.

```
if(iGuard_StartServer(NULL)) {
     printf("Response Server Started.");
}
```

Note: In the above example, the parameter pCallback is set to NULL, which mean the iGuard API Library will not call the application if data is received from iGuard Terminal. Application can set the callback function (i.e. enable the API call the application when data received from iGuard Terminal) by calling function `iGuard_SetRecvAccessLogCallbackFunc()`

PRELIMINARY

iGuard_StartServerWithAutoLog()

Start a Response Server (with Automatically Logging) in local PC to receive the data sent from iGuard terminal.

Function prototype

```
CDECL_DLL(BOOL) iGuard_StartServerWithAutoLog(LPCTSTR
lpzLogFile)
```

Inputs

lpzLogFile pointer to the file name for the iGuard API log incoming access log. NULL to use the default log file name (i.e. access.log in start up directory). Note: File is in comma delimited text format.

Return Value

TRUE if Response Server started successful.

Note: The Response Server is a software module that runs in the background of the PC. It is indicated by an Icon at the system tray (visible of icon can be control by application). It notifies the Master iGuard unit the present of this server, and the Master unit will automatically update the new access log records to the PC. Please refer to the beginning of the document for more details about the Response Server.

Example

The following example shows how to start an iGuard Response Server in local PC.

```
if(iGuard_StartServerWithAutoLog(NULL)) {
     printf("Response Server Started.");
}
```

Note: In the above example, the parameter lpzLogFile is set to NULL, which mean the iGuard API Library will automatically log all the incoming data to a file called "access.log" in the application working directory.

PRELIMINARY

```
iGuard_StopServer()
```

Stop the iGuard Response Server in local PC.

Function prototype
```
CDECL_DLL(BOOL) iGuard_StopServer(void)
```

Inputs
This function does not take any parameter.

Return Value
TRUE if Response Server stop successful.

Example
The following example shows how to stop an iGuard Response Server in local PC.

```
if(iGuard_StopServer()) {
     printf("Response Server Stopped.");
}
```

PRELIMINARY

```
iGuard_RegisterServer()
```

◆ Connect ◆ Master ◆ Login

Register a Response Server to an iGuard master.
Notes: An iGuard master will not send any data to Response Server before it is registered. (Each iGuard Master Terminal support one Response Server only)

Function prototype

```
CDECL_DLL(BOOL) iGuard_RegisterServer(LPCTSTR lpzUrl, BOOL
bLogin)
```

Inputs

lpzUrl	pointer to URL address of the iGuard master.
bLogin	always TRUE (for compatibility).

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to register an iGuard Response Server to an iGuard Terminal.

```
char lpzURL[] = "192.168.0.100";

if(iGuard_Login(lpzURL, "Admin", "123")) {
     if(iGuard_RegisterServer(lpzURL, TRUE)) {
          printf("Response Server Registered.");
     }
}
```

PRELIMINARY

```
iGuard_UnRegisterServer()
```

◆ Connect ◆ Master ◆ Login

Remove the Response Server from iGuard Master Terminal. Refer to iGuard_RegisterServer for details.

Function prototype
CDECL_DLL(BOOL) iGuard_UnRegisterServer(LPCTSTR lpzUrl);

Inputs
lpzUrl pointer to URL address of the iGuard master.

Return Value
TURE if the function is successful. FALSE if the function fails.

Example
The following example shows how to un-register an iGuard Response Server from an iGuard Terminal.

```
char lpzURL[] = "192.168.0.100";

if(iGuard_UnRegisterServer(lpzURL)) {
     printf("Response Server UnRegistered.");
}
```

Note
Refer to iGuard_RegisterServer for details.

PRELIMINARY

iGuard_SetServerPort()

Set the TCP/IP Port No. for receiving data sending from iGuard Terminal.
Note: to keep the changes, call the function `iGuard_SaveConfig()`.

Function prototype

```
CDECL_DLL(BOOL) iGuard_SetServerPort(unsigned short nPort,
BOOL bRestart)
```

Inputs

nPort	TCP/IP Port No.
bRestart	TRUE will restart the Response Server if it's already started.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to change the port of an iGuard Response Server.

```
if(iGuard_SetServerPort(3080, TRUE)) {
     printf("Response Server Port Changed.");
}
```

PRELIMINARY

```
iGuard_GetServerPort()
```

Get the current TCP/IP Port No. assigned for the Response Server.

Function prototype

```
CDECL_DLL(unsigned short) iGuard_GetServerPort(void)
```

Inputs

This function does not take parameter.

Return Value

TURE if the function is successful. FALSE if the function fails.

Example

The following example shows how to get the port of an iGuard Response Server.

```
printf("Response Server Port No. is %d.",
     iGuard_GetServerPort());
```

PRELIMINARY

4. Remote Access Module

Remote access module let application connect to remote network's RAS server easily through the MS Windows' RAS (or Dial up network).

Notes: Remote Access Service is Windows build-in service. iGuard API's RAS function only provide a easier interface with the RAS server for Application Programmers. Programmers can directly access the service. For details information, please refer to the Windows SDK manual.

PRELIMINARY

iGuard_RasInitialize()

iGuard_RasInitialize() initialize the iGuard API's Remote Access Service (need the MS Windows' RAS/Dialup network)

Function prototype

```
CDECL_DLL(BOOL) iGuard_RasInitialize(BOOL bShowMenu);
```

Inputs

bShowMenu Show RAS Phone book on iGuard Tray Icon Menu (by calling iGuard_ShowServerIcon function.)

Return Value

TURE if the function is successful. FALSE if the function fails.

Notes

To use RAS functions of iGuard API, the MS Windows' RAS (NT) or Dialup networking (Win95/98)

PRELIMINARY

`iGuard_RasDialUp()`

Connect to remote network with Dial up network. Notes: The RasDialup is an asynchronous function, i.e. the function will return immediately after setting up the dial up, no matter the connection is success or failed. To check the connection result, you need to call the RASIsConnected function.

Function prototype

```
CDECL_DLL(BOOL) iGuard_RasDialUp(LPCTSTR lpzEntryName,
LPCTSTR lpzUserName, LPCTSTR lpzPassword, BOOL bShowDialog)
```

Inputs

lpzEntryName	pointer to the entry name in RAS phone book
lpzUserName	pointer to the login user name
lpzPassword	pointer to the login password
bShowDialog	TRUE will show the connection progress dialog on screen.

Return Value

TURE if the connection is successful. FALSE if the connection fails.

PRELIMINARY

`iGuard_RasPhoneBook()`

Retrieve the RAS phone book.

Function prototype

```
CDECL_DLL(BOOL) iGuard_RasPhoneBook(LPTSTR lpzPhoneBook,
int nMaxLen)
```

Inputs

lpzPhoneBook	pointer to the buffer to receive the RAS phone book
nMaxLen	maximum length of buffer

Return Value

TURE if the function is successful. FALSE if the function fails.

PRELIMINARY

`iGuard_InsertRasPhoneBookMenu()`

Insert the RAS phone book to a menu. Application need to handle the ID and windows message for the menu's event.

Function prototype

```
CDECL_DLL(BOOL) iGuard_InsertRasPhoneBookMenu(HMENU
hSubMenu, int nPos, int nID)
```

Inputs

hSubMenu	handle to menu
nPos	position of RAS phone book to be inserted
nID	ID of first phone book entry (add one for each item)

Return Value

TURE if the function is successful. FALSE if the function fails.

PRELIMINARY

`iGuard_RasSetLogin()`

Set the login information for RAS connection.

Function prototype

```
CDECL_DLL(BOOL) iGuard_RasSetLogin(LPCTSTR lpzEntryName,
LPCTSTR lpzUserName, LPCTSTR lpzPassword)
```

Inputs

lpzEntryName	pointer to the Phone book entry name
lpzUserName	pointer to the user name
lpzPassword	pointer to the password

Return Value

TURE if the function is successful. FALSE if the function fails.

PRELIMINARY

`iGuard_RasIsConnected()`

Set the login information for RAS connection.

Function prototype
`CDECL_DLL(BOOL) iGuard_RasIsConnected()`

Inputs
This function does not take parameter.

Return Value
TURE if the RAS Connected. FALSE if not connected.

PRELIMINARY

Appendix A. iGuard API Error Codes

The error codes are described below for all functions in the iGuard API Library which can be retrieved by iGuard_GetLastError() function.

Error Code	Definition
0000	Unknown / No error.
1001	Cannot resolve URL address.
1002	WINSOCK Startup Failed.
1003	Error Opening socket.
1004	Connect failed.
1005	Send failed.
1006	Receive failed.
1007	Cannot find specified iGuard in the network.
1008	Unauthorized access.
1009	Cannot create temporary file.
1010	Not a Master.
1011	Cannot create message window.
1012	Server already started.
1013	Response Server startup failed.
1014	Operation cancelled by user.
1015	RAS not initialized.
1016	TCP/IP Port open failed
1017	Invalid TCP/IP Port No.
2001	Out of buffer space.
2002	Save file cancelled.
2003	Save file failed.
2004	Tray Icon not created.
3001	Buffer Size error.
4001	Cannot delete Access Log record.
4002	Unknown database type.
4003	Database with no options.
4004	Read only record.
4005	Function not supported by this database.
4006	BOF
4007	EOF
5001	Invalid URL Address.
5002	Login in setup failed.
6001	Response Server already started by other application.

PRELIMINARY

Appendix B. Examples & Source Codes

Example source code illustrates the use of each of the software modules. Programs are located in the directory, C:\iGuard\examples. Most questions on how to use the DLLs will be answered by these examples.

exRspSvr.c

This source code (project file for Microsoft Visual C++ 6.0 included) illustrates the use of API functions to initialize the API library, check information of connected iGuard (firmware version, serial no.), and receive access log from iGuard automatically (Response Server).

Monitor.exe

Application to demonstrate the function of iGuard API's Response Server. The access log will display on the application's window and log to a file called "access.log" in the application's working directory. Source code not available at that moment, will be released in later version.

More source code examples will be available later. Please visit our web site for update.

http://www.iguardsystem.com

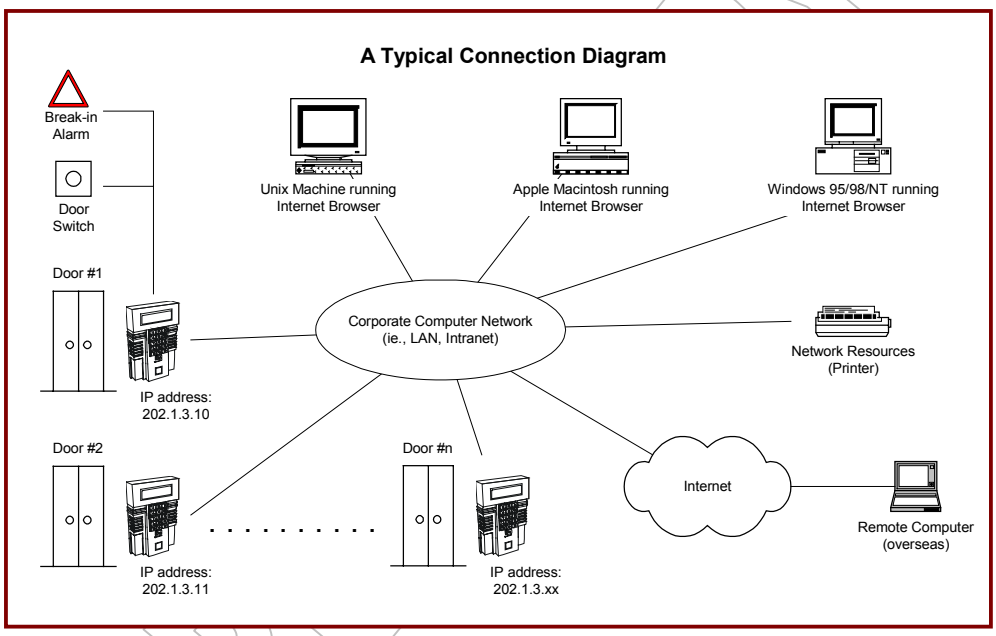
PRELIMINARY

Appendix C. Network Connections

To use the iGuard API to access / control iGuard Terminal, you need a network connection. This appendix shows you how to connect the iGuard with your network / personal computer.

Typical Connection

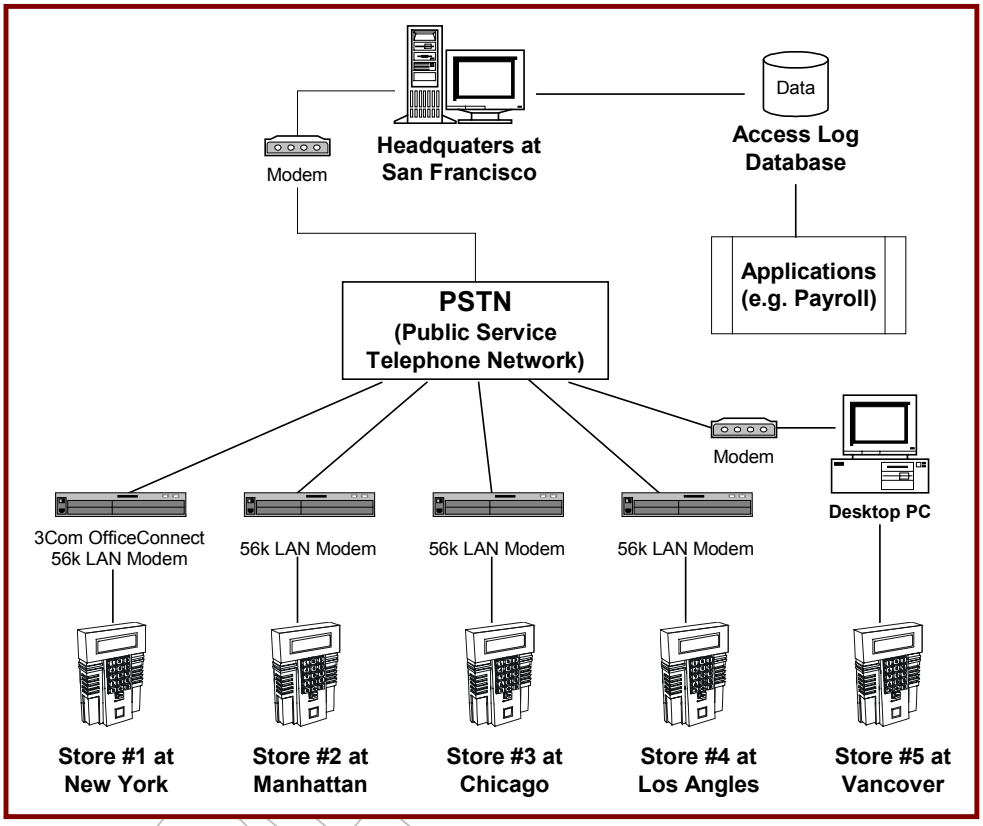
Here is the typical connection diagram. All you need is connecting iGuard terminal to your computer network with a Ethernet cable. Then all the computer in network can access iGuard Terminals. Besides, with iGuard's Master / Slave Technology, information can easily share between iGuard Terminals. See iGuard operation manual for details. With this connection method, all the access record will be stored in the iGuard Master Terminal.



Central Office – Remote Site Connection

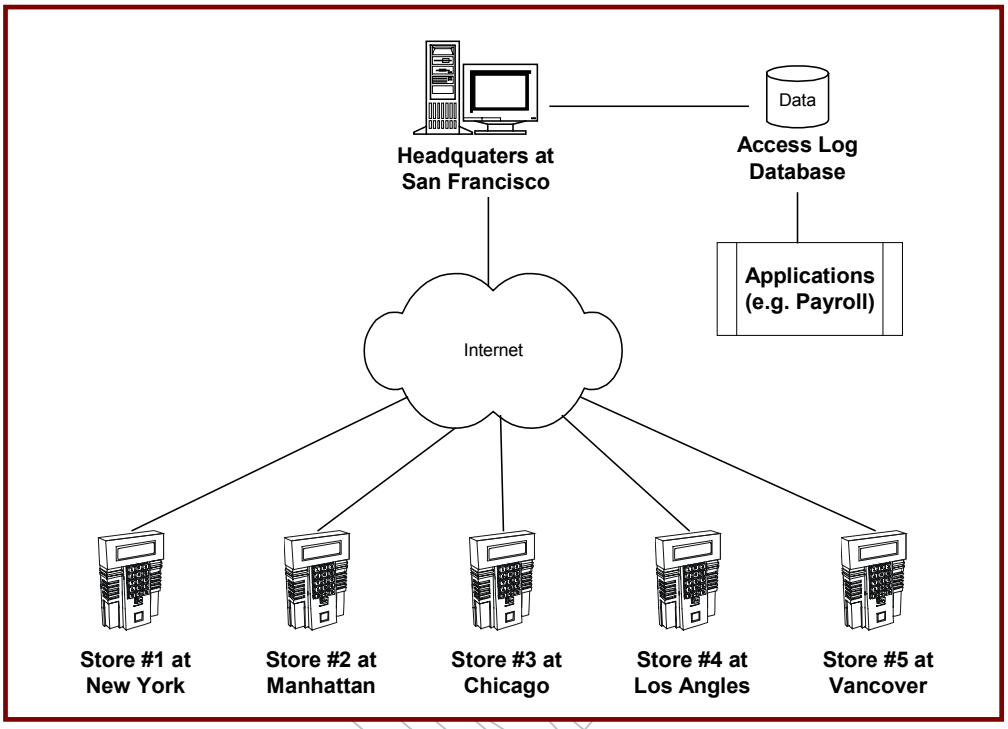
1. With Modems

With iGuard API, your application can connect and collect data from remote site's iGuard Terminals through the Microsoft Windows' (95/98/NT) Remote Access Service (RAS).



2. With Internet

If remote site and head office both have a direct connection (for example lease line connection) to the Internet. Then the iGuard API can directly collecting data of iGuard Terminals in remote site through the Internet without any dial up.



PRELIM