GuardPointPro APIs

V 1.2

Publication No. 10UExxx Issue: July, the 30th 2008

TABLE OF CONTENTS

Introduction

This document is dedicated to explain the existing API of GuardPointPro

It allows an **easy integration** with SENSOR ACCESS Access control and alarm monitoring software called GuardPointPro.

This means that an external application could

- receive many information from GuardPointPro such as online events of access control system's (Access granted, Access denied, Start of Alarm, …)
- and act on Access control system's by
    - Creating cardholders
    - Manages doors status, and relays status (Open a door for a while, open constantly, close constantly, or return to default status)
    - Manage alarm status (disarm a zone / input group)
    - Executing existing actions and processes of GuardPointPro
    - Login / Logoff
    - User interface (messages on screen)
    - Download configuration to controllers (that may be updated directly in DB by an external application)

GuardPointPro has also other integration gateway such as

-
- OPC
- ModbusTCP
- Wizcon

The document is based on GuardPointPro Version 1.8.003 (June 2008)
Most of the commands are supported in previous versions, but in order to simplify we will only work on the basis of the actual version.
To get the latest version of GuardPointPro consult
http://www.sensoraccess.co.uk/

The communication with GuardPointPro is done by a communication engine called "Spread". For more information about Spread, see www.spread.org

Installation

Unzip the zip file.

Install in your project folder the acAPI.dll.

Add Reference to this DLL in your project.

Sample Program

The acAPI.dll is given with a sample program, including its source code, named Test_AcAPI.exe

This project Test_AcAPI was compiled with Visual Basic 2008 (with compatibility with .NET Framework v2.0). The source code of this program is delivered in the zip file.

Have a look at the program source code to help you to fast interface with the acAPI.dll.

Properties

DestinationPCName

This property defines the PC we are sending it the commands.

Default value: *<Current PC Name>*

SpreadConfiguration

This property defines how to connect to Spread (Communication layer) that we use to send the commands.

Default value: *4803@localhost*

Methods

ChangeUserLogin

Send a request to change the user logged in by another one with the user name and password.

Syntax:

```
Sub ChangeUserLogin(ByVal User As String, ByVal Password As String)
```

Sample:

```
acDLL.ChangeUserLogin("user", "password")
```

OpenScreen

Send a request to open a screen.
The command supports selecting

- on which record,
- on which tab
- and the screen size (Normal, minimize, or maximize).

Syntax:

```
Sub OpenScreen(ByVal Screen As String, Optional ByVal MinMax As
Integer = 0, Optional ByVal ViewPhotoListReader As Long = 0, Optional
ByVal onRecordID As Long = 0, Optional ByVal onTabNumber As Integer =
0)
```

Sample:

```
acDLL.OpenScreen("ID_Cardholders")
```

Cf Appendix A (Screens ID) to get all the parameter according to the screen you want to open.

PreviewVideo

Send a request to preview a camera live video with the db_CameraID (cf Appendix B).

Syntax:

```
Sub PreviewVideo(ByVal db_CameraID As Long)
```

Sample:

```
acDLL.PreviewVideo(1)
```

PreviewReport

Send a request to preview an existing report with the report full name.

Syntax:

```
Sub PreviewReport(ByVal ReportFile As String)
```

Sample:
acDLL.PreviewReport("C:\Program Files\GuardPointPro\Reports\Last report.rpx")

DisplayMessage

Send a request to display a message box with the text.

Syntax:
```
Sub DisplayMessage(ByVal Message As String)
```

Sample:
```
acDLL.DisplayMessage("Hi, How are you? ")
```

InsertTextinLog

Send a request to insert message in the Log windows.

Syntax:
```
Sub InsertTextinLog(ByVal Text As String)
```

Sample:
```
acDLL.InsertTextinLog("Hi, How are you? ")
```

InsertTextinStatusBar

Send a request to insert message in the status bar and set the percent of the progress bar.

Syntax:
```
Sub InsertTextinStatusBar(ByVal Text As String, ByVal Percent As Byte)
```

Sample:
```
acDLL.InsertTextinStatusBar("Downloading Cardholders", 30)
```

PlaySound

Send a request to play a sound file with the full path of the sound file.

Syntax:
```
Sub PlaySound(ByVal SoundFile As String)
```

Sample:
```
acDLL.PlaySound("C:\Windows\Media\Windows Notify.wav")
```

ExecuteAction

Send a request to execute an existing action with db_ActionID (cf Appendix B).

Syntax:
```
Sub ExecuteAction(ByVal db_ActionID As Long)
```

Sample:
```
acDLL.ExcecuteAction(1)
```

ExecuteProcess

Send a request to preview an existing process with db_ProcessID (cf Appendix B)

Syntax:
```
Sub ExecuteProcess(ByVal db_ProcessID As Long)
```

Sample:
```
acDLL.ExecuteProcess(1)
```

ExecuteApplication

Send a request to execute an application file with the full path of the application file.

Syntax:
```
ExecuteApplication(ByVal ApplicationPath As String)
```

Sample:
```
acDLL.ExecuteApplication("C:\Windows\Calc.exe")
```

InitializeController

Send a request to initialize an existing controller db_ControllerID (cf Appendix B).

Syntax:
```
Sub InitializeController(ByVal db_ControllerID As Long, Optional ByVal
WithoutCardholder As Boolean = False, Optional ByVal EraseOption As
EraseOptions = EraseOptions.AllDB_NoBuffer)
```

Where EraseOptions are:

- EventBufferOnly
- AllDBExceptCardholder
- CardholderOnly
- EventBuffer_And_AllDBExceptCardholder
- EventBuffer_AndCardholder
- AllDB_NoBuffer
- ALL_memory

RecreateMemoryTables

Send a request to initialize an existing controller db_ControllerID (cf Appendix B) with recreation of memory tables.

Syntax:
```
Sub RecreateMemoryTables(ByVal db_ControllerID As Long)
```

SetRelayState

Send a request to modify the relay state (Activate the relay / Inhibit the relay) with the db_OutputID (cf Appendix B).

Syntax:
```
Sub SetRelayState(ByVal db_OutputID As Long, ByVal RelayState As
RelayStates, Optional ByVal Delay As Integer = 0)
```

Where RelayStates are:

- Normal
- ConstantON
- ConstantOFF
- Delay

Sample: To activate the relay 1 Constant ON
```
acDLL.SetRelayState(1, acAPI.API.RelayStates.ConstantON)
```

This command allows to control doors relays and other output (e.g. alarm siren)

SetInputState

Send a request to modify the Input state (Deactivate / Force activate the input) with the db_InputID (cf Appendix B).

Syntax:
```
Sub SetInputState(ByVal db_InputID As Long, ByVal InputState As
InputStates, Optional ByVal Delay As Integer = 0)
```

Where InputStates are:

- Normal
- ConstantActivate
- ConstantDeactivate
- Delay

Sample: to return to normal mode the input 1
```
acDLL.SetInputState(1,acAPI.API.InputStates.Normal)
```

This command allows to control alarms sensors to be arm or not.

SetInputGroupState

Send a request to modify the Input group state (Deactivate / Force activate the input group) with the db_InputGroupID (cf Appendix B).

Syntax:
```
Sub SetInputGroupState(ByVal InputGroupID As Long, ByVal
InputGroupState As InputGroupStates, Optional ByVal Delay As Integer =
0)
```

Where InputGroupStates are:

- Disarm_DuringSeconds
- Disarm_DuringMinutes
- Disarm_Constantly
- Disarm_UntilTimeZone
- Disarm_NONE_CancelPreviousDelay
- Arm_DuringSeconds
- Arm_DuringMinutes
- Arm_Constantly
- Arm_UntilTimeZone
- Arm_NONE_CancelPreviousDelay

Sample: To disarm the input group 1 during 30 seconds
```
acDLL.SetInputGroupState(1,
acAPI.API.InputGroupStates.Disarm_DuringSeconds, 30)
```

This command allows to control alarm zones (defined as group of inputs) to be arm or not.

ActivateAllDoorRelays

Send a request to modify the Input state (Deactivate / Force activate the input) with the db_InputID (cf Appendix B).

Syntax:
```
Sub ActivateAllDoorRelays(ByVal db_ControllerID As Long, ByVal
DoorRelayState As DoorRelayStates, Optional ByVal Delay As Integer =
0)
```

GetTimeDate

Send a request to get the time and date of a controller db_ControllerID (cf Appendix B).

Syntax:
```
Function GetTimeDate(ByVal db_ControllerID As Long) As Date
```

GetDigitalInputStatus

Send a request to get the input and output status of a controller db_ControllerID (cf Appendix B).
It returns the logical state of the input (physical state according to NO/NC) and the

Syntax:
```
Function GetDigitalInputStatus(ByVal db_ControllerID As Long) As
String
```

The returned string is build of 0/1 in the following order

- Inputs (1-16)
- Relays (1-64)
- Inputs (17-24)

GetHardwareVersion

Send a request to get the hardware version of a controller db_ControllerID (cf Appendix B).
It returns a string. For more information, consult the TPL User Manual.

Syntax:

```
Function GetHardwareVersion(ByVal db_ControllerID As Long) As String
```

GetFirmareVersion

Send a request to get the firmware version of a controller db_ControllerID (cf Appendix B).
It returns the Eprom date and checksum.

Syntax:

```
Function GetFirmareVersion(ByVal db_ControllerID As Long) As String
```

GetMemoryOccupation

Send a request to get the memory occupation of a controller db_ControllerID (cf Appendix B).
It returns the number of cardholders stored in the controller memory.

Syntax:

```
Function GetMemoryOccupation(ByVal db_ControllerID As Long) As Long
```

isPollingNow

Send a request to know if currently we are polling or not the controllers.
It returns True/False.

Syntax:

```
Function isPollingNow() As Boolean
```

StartPolling

Send a request to Start Polling the controllers.
This command update the polling queues, it adds new controllers or remove controllers have been set as not active.
You can specify a specific controller or network. Without defining any controller, it stops all the communication polling with the controllers.

Syntax:

```
Sub StartPolling(Optional ByVal db_ControllerID As Long = 0, Optional
ByVal db_NetworkID As Long = 0)
```

StopPolling

Send a request to Stop Polling the controllers.
You can specify a specific controller. Without defining any controller, it stops all the communication polling with the controllers.

Syntax:

```
Sub StopPolling(Optional ByVal db_ControllerID As Long = 0)
```

ImportCardholder

Send a request to import a cardholder in the database and inform the controllers. This allows adding, updating or deleting cardholders.

Syntax:

```
Function ImportCardHolder(ByVal a As CardholderFields) As
ImportResults
```

Where **CardholderFields** contains the cardholder fields.

Properties of **CardholderFields**
`Number As String`
`Last_Name As String`
`First_Name As String`

`Type As CardholderTypes`
`Visitor`
`Cardholder`
`Guard`
`DELETED`

`Badge As String`

`Technology As CardTechnologies`
`Magnetic`
`BarCode`
`Wiegand`
`Wiegand2`
`WiegandKeypad`
`BioSmartCard`
`Touch`
`Radio`

`Photo As String`
`Department As String`
`Office_Phone As String`
`Access_Group As String`
`PIN_code As String`
`From_Date As String`
`To_Date As String`
`Validated As Boolean (Default value True)`
`Street As String`
`City As String`
`ZIP As String`
`Personal_Phone As String`
`Description As String`
`Car_Number As String`
`ID As String`

```
Supervisor As Boolean
Label_1 As String
Label_2 As String
Label_3 As String
Label_4 As String
Company As String
Lift_Program As String
Parking_Users_Group As String
MultiSite_Type As MultiSite_Types
     isLocal
     isShared
     isGlobal
Site As String
Personal_WP As String
Personal_CL As String
Keep_card_on_motorized_reader As Boolean
No_APB As Boolean
No_access_during_holidays As Boolean
Reset_APB As Boolean
Need_Escort As Boolean
Badge_Printing_Layout As String
Visited_person As String
Visited_person_location As String
Visit_purpose As String
```

And **ImportResults** options are:

- UpdateSuccessfully
- InsertSuccessfully
- MandatoryFieldMissing
- UpdateFailed
- InsertFailed
- AuthorisationExcedded
- CannotChangeGuard
- DuplicateName
- CardHolderDeleted
- BadgeCodeNotOK

The import creates

- the cardholder,
- the badge,
- the access group if not found,
- the department if not found,
- the lift program if not found,
- the parking user group if not found,
- the personal weekly program if not found

It supports

- Multiple Access Group (use ; to separate the names of the access group)
- Dynamic Fields
- Multi site fields

For more details about the import, consult the user manual of GuardPointPro about import profiles.

Events

AccessEvent

Wake up the application when an Access event arrives.

We return an object **AccessEventRecord** that contains:

Properties of **AccessEventRecord**	
`EventDate As Date`	Event description

`EventType As EventTypes`
`Access Granted`
`Access Granted with Duress Code`
`Access Denied`
`Access Denied Too Much Trials`
`Unknown Badge`
`Unknown Badge Too Much Trials`
`Non Allocated Badge`	

`ReaderName As String`	
`TransactionCode As Integer`	
`CardHolderName As String`	
`CardHolderPhotoFileName As String`	
`Denied_WrongFinger As Boolean`	
`Denied_WrongKeypadCode As Boolean`	
`Denied_FullorLock As Boolean`	
`Denied_Time As Boolean`	
`Denied_APB As Boolean`	
`Denied_ReaderNotAllowed As Boolean`	
`Denied_SiteCode As Boolean`	
`Denied_InhibitedCardholder As Boolean`	
`Denied_AccessGroup As Boolean`	
`isEscort As Boolean`	
`Denied_EscortTimeout As Boolean`	
`Denied_EscortNotAuthorized As Boolean`	
`db_ReaderID As Long`	Database Fields reference Cf Appendix B
`db_ControllerID As Long`	
`db_CardHolderID As Long`	
`db_ReaderSocID As Long`	
`db_CardHolderSocID As Long`	
`db_CameraID As Long`	
`db_TableLOG_ID As Long`	

Sample:

```
Dim txt As String
txt = AccessRec.EventDate & " "
If AccessRec.EventType = acAPI.AccessEventRecord.EventTypes.AccessGranted Then
       txt = txt & " Access Granted "
End If
txt = txt & AccessRec.CardHolderName & " From Reader " & AccessRec.ReaderName
txt = txt & " Transaction Code " & AccessRec.TransactionCode
Me.RichTextBox1.SelectionColor = Color.Green
Me.RichTextBox1.AppendText(txt & vbCrLf)
```

For more code sample, see function `Private Sub acDLL_AccessEvent(ByVal AccessRec As acAPI.AccessEventRecord) Handles acDLL.AccessEvent`

AlarmEvent

Wake up the application when an Alarm event arrives.

We return an object **AlarmEventRecord** that contains:

Properties of **AlarmEventRecord**	
`EventDate As Date`	Event description

`EventType As EventTypes`
`StartOfAlarm_Immediate`
`StartOfAlarm_Delayed`
`EndOfAlarm`
`LineShort`
`LineCut`
`Status1_AnalogInput`
`Status2_AnalogInput`
`Status3_AnalogInput`
`Status4_AnalogInput`	

`isFromBus2 As Boolean`	
`isDoorContact As Boolean`	
`isRTX As Boolean`	
`DoorName As String`	
`db_InputID As Long`	Database Fields reference Cf Appendix B
`db_ControllerID As Long`	
`db_SocID As Long`	
`db_CameraID As Long`	
`db_TableLOG_ID As Long`	

Sample:

```
Dim txt As String
txt = AlarmRec.EventDate & " "
Select Case AlarmRec.EventType
Case acAPI.AlarmEventRecord.EventTypes.StartOfAlarm_Immediate
      txt = txt & " Start of Alarm Immediate "
Case acAPI.AlarmEventRecord.EventTypes.StartOfAlarm_Delayed
      txt = txt & " Start of Alarm Delayed "
Case acAPI.AlarmEventRecord.EventTypes.EndOfAlarm
      txt = txt & " End Of Alarm "
End Select

txt = txt & " From Input " & AlarmRec.InputName
Me.RichTextBox1.AppendText(txt & vbCrLf)
```

For more code sample, see function `Private Sub acDLL_AlarmEvent(ByVal AlarmRec As acAPI.AlarmEventRecord) Handles acDLL.AlarmEvent`

TechnicalEvent

Wake up the application when a technical event arrives.

We return an object **TechnicalEventRecord** that contains:

Properties of **TechnicalEventRecord**	
`EventDate As Date`	Event description

`EventType As EventTypes`
`TableError`
`LowBattery`
`PowerDown`	

PowerUp 'For MEGA Only PowerSupplyFailure PowerSupplyOK BoxOpened BoxClosed ReaderDisconnected ReaderConnected	
ReaderName As String	
ControllerName As String	
db_ControllerID As Long	Database Fields reference Cf Appendix B
db_ReaderID As Long	
db_SocID As Long	
db_TableLOG_ID As Long	

Sample:

```
Dim txt As String
txt = TechRec.EventDate & " "
Select Case TechRec.EventType
Case acAPI.TechnicalEventRecord.EventTypes.PowerUp
      txt = txt & " PowerUp "
Case acAPI.TechnicalEventRecord.EventTypes.PowerDown
      txt = txt & " Power Down "
End Select

txt = txt & " From Controller " & TechRec.ControllerName
Me.RichTextBox1.AppendText(txt & vbCrLf)
```

For more code sample, see function `Private Sub acDLL_TechnicalEvent(ByVal TechRec As acAPI.TechnicalEventRecord) Handles acDLL.TechnicalEvent`

UserEvent

Wake up the application when a user event arrives.

We return an object **UserEventRecord** that contains:

Properties of **UserEventRecord**	
EventDate As Date	Event description
EventType As EventTypes NewRecord SaveRecord DeleteRecord Login Logout	
UserName As String	
WorkStationName As String	
RecordScreen As String	
RecordDetails As String	
db_ UserID As Long	Database Fields reference Cf Appendix B
db_SocID As Long	
db_TableLOG_ID As Long	

Sample:

```
Dim txt As String
txt = anUserEventRecord.EventDate & " "
Select Case anUserEventRecord.EventType
Case acAPI.UserEventRecord.EventTypes.Logout
```

```
        txt = txt & " Logout From user " & anUserEventRecord.UserName &
" - " & anUserEventRecord.WorkStationName
Case acAPI.UserEventRecord.EventTypes.SaveRecord
        txt = txt & " Save Record by user " & anUserEventRecord.UserName
& " - " & anUserEventRecord.RecordScreen & " " &
anUserEventRecord.RecordDetails
End Select
Me.RichTextBox1.AppendText(txt & vbCrLf)
```

For more code sample, see function `Private Sub acDLL_UserEvent(ByVal anUserEventRecord As acAPI.UserEventRecord) Handles acDLL.UserEvent`

ControllerCommunicationError

Wake up the application when a controller starts to be in Communication error.
It returns the text to be displayed in Log windows.

ControllerCommunicationOK

Wake up the application when a controller returns to be in Communication OK.
It returns the text to be displayed in Log windows.

TextForLog

Wake up the application when information to be displayed in Log windows arrives.
It returns the text to be displayed in Log windows.

TextForStatusBar

Wake up the application when information to be displayed in Status bar arrives.
It returns the text to be displayed in the Status bar and the percent of the progress bar.

This is useful to get the feedback of the progress of the initialization of a controller.

APInotDocumented

Wake up the application when a message arrives that has not been documented in the current API.

For specific integration request that has not been documented here, please contact us.

SpreadPollingError

Wake up the application when the communication layer Spread returns an error code.

The error codes are:

```
-1    ILLEGAL_SPREAD
-2    COULD_NOT_CONNECT
-3    REJECT_QUOTA
-4    REJECT_NO_NAME
-5    REJECT_ILLEGAL_NAME
-6    REJECT_NOT_UNIQUE
-7    REJECT_VERSION
-8    CONNECTION_CLOSED
-9    REJECT_AUTH
-11   ILLEGAL_SESSION
-12   ILLEGAL_SERVICE
-13   ILLEGAL_MESSAGE
-14   ILLEGAL_GROUP
-15   BUFFER_TOO_SHORT
-16   GROUPS_TOO_SHORT
-17   MESSAGE_TOO_LONG
```

The more frequents error codes are -8 and -11 and happens when the spread process is not running (most of the time due to a wrong configuration file spread.conf). Check the Computers definition in GuardPointPro.

ExceptionHappen

Wake up the application when an exception happens in the DLL code.

It returns the Function Name and the exception object.

Appendix A: Screens ID

Screens ID	Description
ID_APBLevel	Anti Pass Back Level
ID_Area	Area
ID_Departement	Department
ID_Diagnostic	Diagnose
ID_Visitor	Visitor
ID_AccessGroup	Access Group
ID_Actions	Action
ID_Badge	Badge
ID_Cardholders	All Cardholders
ID_Computer	Computer
ID_Configuration	Customized Label
ID_Controllers	Controller
ID_Counters	Counter
ID_DailyProgram	Daily Program
ID_EventHandlingProgram	Event Handling Program
ID_GlobalReflex	Global Reflex
ID_InputGroup	Input Group
ID_OutputGroup	Output Group
ID_Holiday	Holiday
ID_Log	Active Alarms
ID_Network	Network
ID_Process	Process
ID_WeeklyProgram	Weekly Program
ID_ParkingDefinition	Parking Lot
ID_Company	Company / Site
ID_ZoneID	Parking User Group
ID_User	Users
ID_AuthorisationsLevels	Authorisation Levels
ID_Icons	Icons / Symbols
ID_Maps	Maps
ID_Positions	Position
ID_LiftAuthorisationGroups	Lift Authorisation group (only when Lift per Reader)
ID_LiftProgram	Lift program
ID_TimeAttendance	Roll Call
ID_CrisisLevel	Send a Crisis Level
ID_ExecuteProcess	Execute Process
ID_GuardDefinition	Guard Definition
ID_ViewPhoto	View Photo
ID_PatrolTour	Patrol Tour
ID_CheckPoint	Checkpoints
ID_PatrolStatus	Patrol status
ID_DisplayJournalSmall	Report wizard
ID_CreateagroupofBadges	Group of Badge
ID_ImportProfile	Import profiles
ID_CustomizedFields	Customized fields
ID_Camera	Camera
ID_Matrix	Matrix
ID_LocationStatus	Location Status

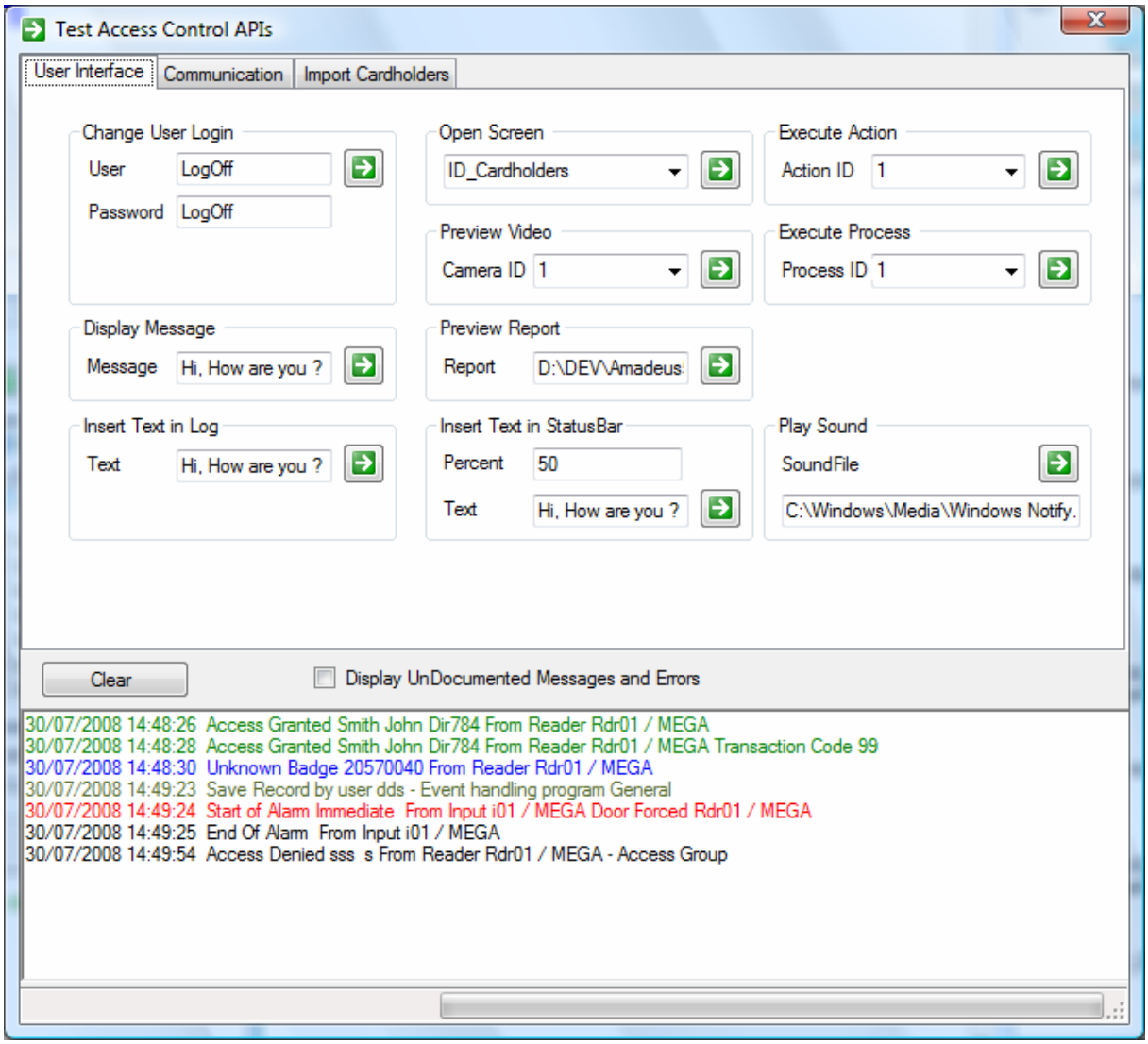
Appendix B: Database Fields

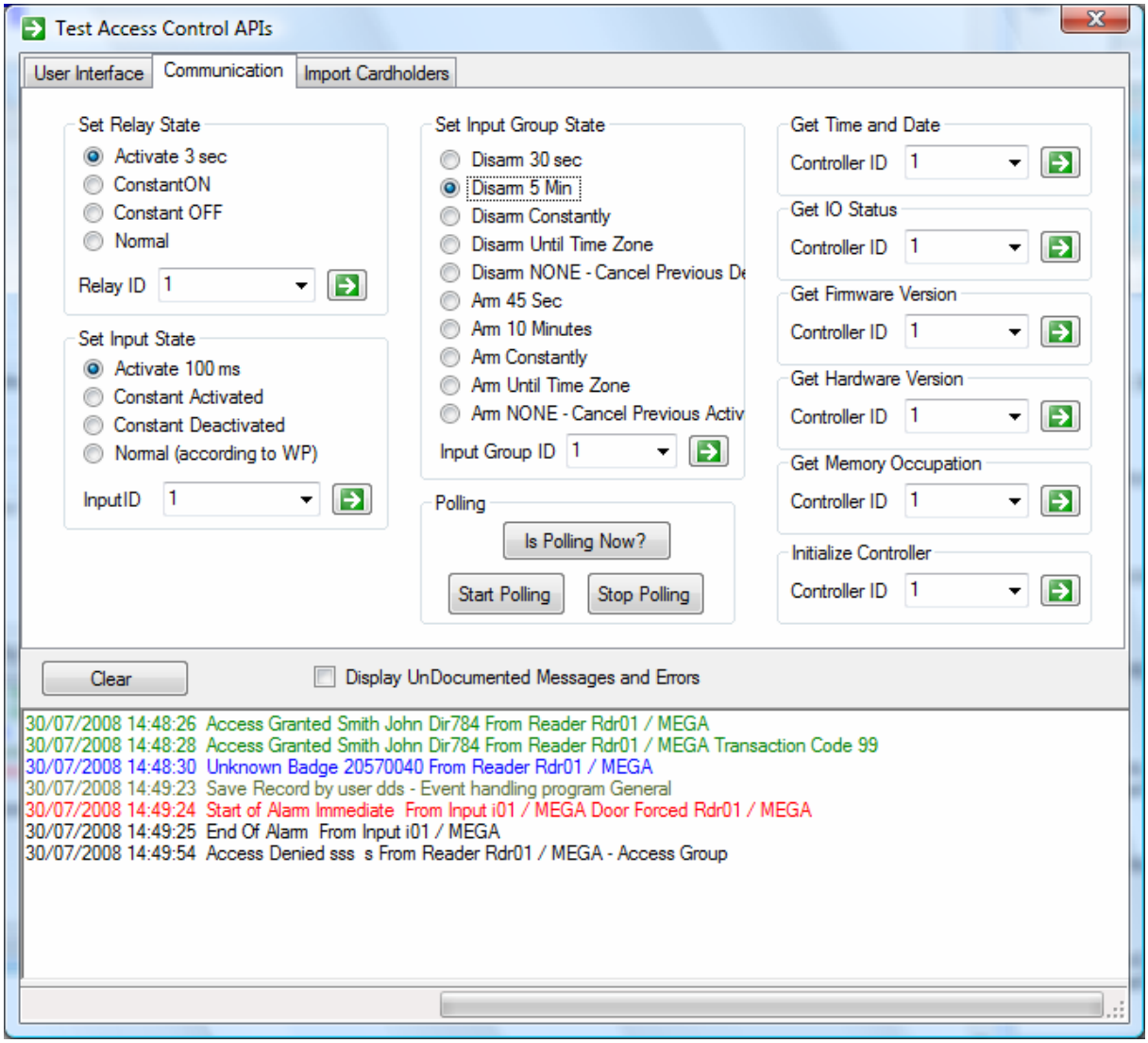
The database fields

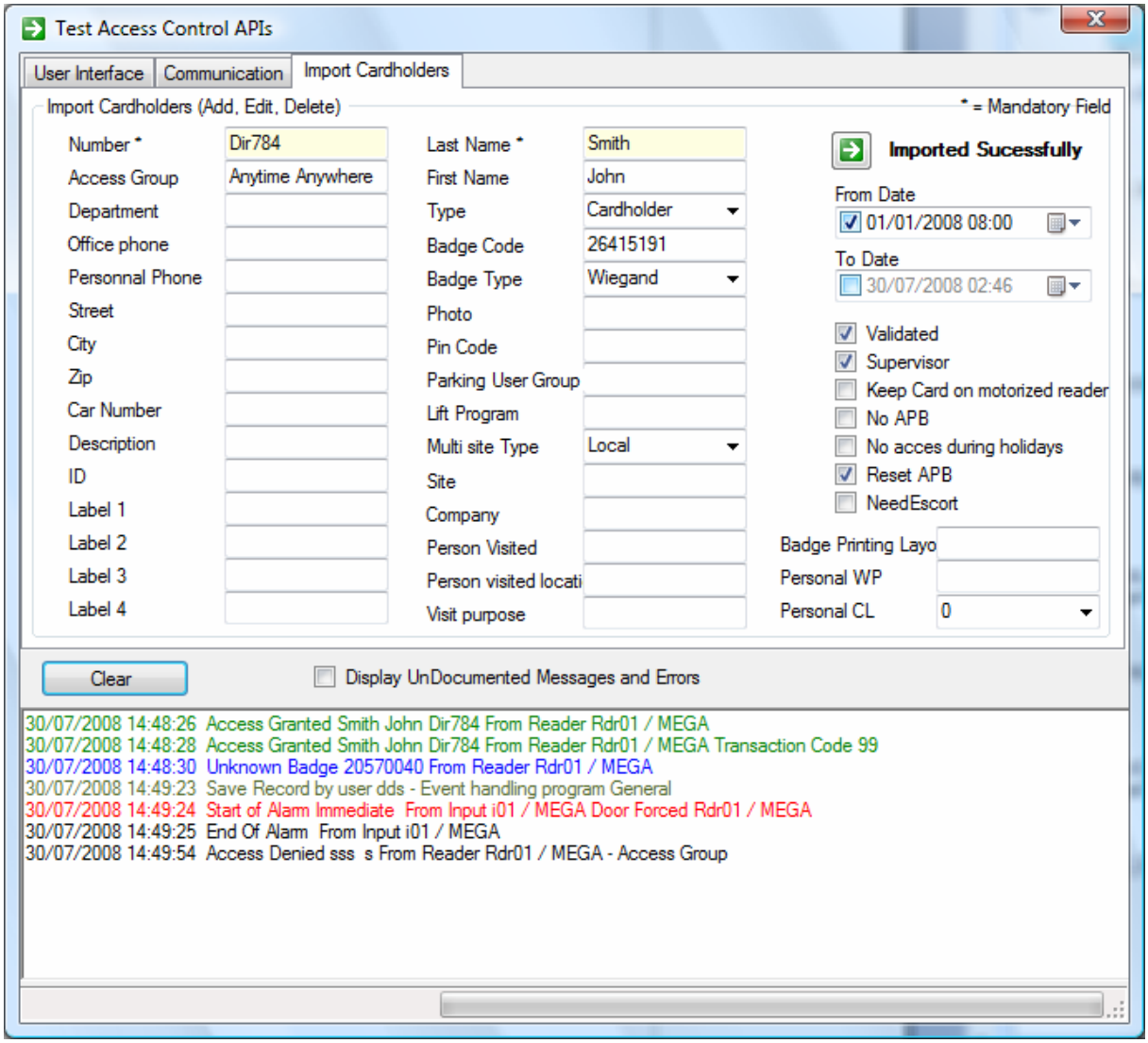
db_ControllerID	Select ID, Name from Controller
db_ReaderID	Select ID, Name from Reader
db_InputID	Select ID, Name from [Input]
db_OuputID	Select ID, Name from [Output]
db_NetworkID	Select ID, Name from Network
db_SocID db_ReaderSocID db_CardHolderSocID	Select ID, Name from SOC
db_TableLOG_ID	Select ID from LOG
db_CardHolderID	Select ID, Last_Name & ‘ ‘ & First_Name as Name from CRDHLD
db_CameraID	Select ID, Name from Camera
db_InputGroupID	Select ID, Name from IGrp
db_ActionID	Select ID, Name from [Action]
db_ProcessID	Select ID, Name from Process

We plan to improve the API to allow access to Database tables and records.

Appendix C: Sample application print screen




Test Access Control APIs
User Interface
Communication
Import Cardholders
Set Relay State
Activate 3 sec
ConstantON
Constant OFF
Normal
Relay ID 1
Set Input State
Activate 100 ms
Constant Activated
Constant Deactivated
Normal (according to WP)
InputID 1
Set Input Group State
Disarm 30 sec
Disarm 5 Min
Disarm Constantly
Disarm Until Time Zone
Disarm NONE - Cancel Previous De
Arm 45 Sec
Arm 10 Minutes
Arm Constantly
Arm Until Time Zone
Arm NONE - Cancel Previous Activ
Input Group ID 1
Polling
Is Polling Now?
Start Polling
Stop Polling
Get Time and Date
Controller ID 1
Get IO Status
Controller ID 1
Get Firmware Version
Controller ID 1
Get Hardware Version
Controller ID 1
Get Memory Occupation
Controller ID 1
Initialize Controller
Controller ID 1
Clear
Display UnDocumented Messages and Errors
30/07/2008 14:48:26 Access Granted Smith John Dir784 From Reader Rdr01 / MEGA
30/07/2008 14:48:28 Access Granted Smith John Dir784 From Reader Rdr01 / MEGA Transaction Code 99
30/07/2008 14:48:30 Unknown Badge 20570040 From Reader Rdr01 / MEGA
30/07/2008 14:49:23 Save Record by user dds - Event handling program General
30/07/2008 14:49:24 Start of Alarm Immediate From Input i01 / MEGA Door Forced Rdr01 / MEGA
30/07/2008 14:49:25 End Of Alarm From Input i01 / MEGA
30/07/2008 14:49:54 Access Denied sss s From Reader Rdr01 / MEGA - Access Group


Test Access Control APIs
User Interface
Communication
Import Cardholders
Import Cardholders (Add, Edit, Delete)
* = Mandatory Field
Number *
Dir784
Access Group
Anytime Anywhere
Department
Office phone
Personnal Phone
Street
City
Zip
Car Number
Description
ID
Label 1
Label 2
Label 3
Label 4
Last Name *
Smith
First Name
John
Type
Cardholder
Badge Code
26415191
Badge Type
Wiegand
Photo
Pin Code
Parking User Group
Lift Program
Multi site Type
Local
Site
Company
Person Visited
Person visited locati
Visit purpose
Imported Sucessfully
From Date
01/01/2008 08:00
To Date
30/07/2008 02:46
Validated
Supervisor
Keep Card on motorized reader
No APB
No acces during holidays
Reset APB
NeedEscort
Badge Printing Layo
Personal WP
Personal CL
0
Clear
Display UnDocumented Messages and Errors
30/07/2008 14:48:26 Access Granted Smith John Dir784 From Reader Rdr01 / MEGA
30/07/2008 14:48:28 Access Granted Smith John Dir784 From Reader Rdr01 / MEGA Transaction Code 99
30/07/2008 14:48:30 Unknown Badge 20570040 From Reader Rdr01 / MEGA
30/07/2008 14:49:23 Save Record by user dds - Event handling program General
30/07/2008 14:49:24 Start of Alarm Immediate From Input i01 / MEGA Door Forced Rdr01 / MEGA
30/07/2008 14:49:25 End Of Alarm From Input i01 / MEGA
30/07/2008 14:49:54 Access Denied sss s From Reader Rdr01 / MEGA - Access Group