Access Professional Edition

API for Cardholder Data

BOSCH

en Software Manual

Table of contents

1 Copyright Information

License Agreement:
You should carefully read the following license agreement before proceeding with this installation. This Agreement is between you and Bosch Sicherheitssysteme GmbH, Postfach 1111, 85626 Grasbrunn ("Bosch"). By continuing with the installation, you indicate your acceptance of the terms of this legal Agreement. If you do not agree to the terms of this Agreement, promptly return this product to Bosch.

Copyright/Proprietary Protection:
This Bosch Access Professional Edition Software (the "Software") and the Documentation (all of the online help files and manuals, and all of the printed material included with this Software) are owned by Bosch and are protected by the German Law and international copyright laws and international treaty provisions. You must treat this software like any other copyrighted material, with the exceptions outlined in the following License Grant. Any violation of this agreement will automatically terminate your right to use this Software, and you must immediately return it to Bosch.

License Grant:
Bosch grants you a nonexclusive license to use this Software. You may copy this Software onto the hard disk of this computer, and make one copy for archival purposes. You may not make copies of the Software for any purpose other than what is stated above. You may not copy the Documentation for any reason. You may not reserve-engineer, disassemble, decompile or attempt to discover the source code of the Software. You may not modify the Software or create derivative products using this Software. You may not sublicense, rent or lease any portion of the Software. You may transfer your rights under this agreement on a permanent basis to another person or entity provided that you transfer this License Agreement, all original and updated Software and Documentation, and that you not retain any copies of the Software or Documentation. You must notify Bosch in writing of your transfer, and the recipient must also agree to the terms of this License Agreement.

No Warranties:
Bosch disclaims all warranties, either expressed or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with respect to this Software and the accompanying Documentation.

Limitation of Liability:
Under no circumstances, including negligence, shall Bosch, its employees, or its suppliers or resellers be liable for any direct, indirect, incidental, special, consequential, or any damages wathsoever (including, without limitation, damages for loss of profits, business interruption, attorneys fees, loss of information or other pecuniary loss) arising out of the use of or inability to use this Bosch product, even if Bosch has been advised of the possibility of such damages. Notwithstanding the foregoing, in no event shall the total liability of Bosch, or its employees, suppliers or resellers, for all damages, losses and causes of action, either jointly or severally, exceed the amount paid by you to Bosch or its resellers in the twelve (12) months prior to the claimed injury or damage.

Miscellaneous:
a. Nothing contained herein shall be deemed to convey to you any title or ownership interest in the Software or intellectual property rights related to such Software.
b. Any failure of Bosch to enforce any of the provisions of this Agreement will not be construed as a waiver of such right of Bosch to enforce each and every such provision.
c. If any provision of this Agreement shall be invalid, the invalid provision shall not affect the validity or enforceability of the remaining provisions of this Agreement.

d. This Agreement constitutes the entire agreement between you and Bosch and supersedes any prior agreement concerning the contents of the disc/diskette envelope. Bosch is not bound by any provision of any purchase order or any other type of correspondence (written or verbal).
e. This Agreement is governed by the laws of the Federal Republic of Germany

Use of the Sample Code:
Use of the Sample Code provided within the API for Cardholder data for the Bosch Access Professional Edition System:
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files, to deal with the Sample Code provided within the API included in the Software without restriction, including the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SAMPLE CODE OF THE SOFTWARE DEVELOPMENT KIT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Microsoft®, Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Other product and company names mentioned herein may be the trademarks of their respective owners.

2 Introduction

The API for Cardholder Data can be used to help you integrate access control functionalities into your applications based on the Access Professional Edition version 3.2 or later.

What is in this guide?

This guide provides detailed descriptions of the APIs that can be used to develop access control applications.

Who should read this guide?

This guide is meant for C# developers who would like to develop access control applications. The API can be used with .NET framework 4.

The demo app as part of the distribution package can be modified with Microsoft® Visual Studio 2010.

3 Overview

The API for Cardholder Data is a reusable kit that can be used by developers to integrate access control functionalities into their applications. The API is based on the Access Professional Edition version 3.2 or later. API integration of access control on 3rd party applications enables the following functionalities:

1. Read all person and card data.
2. Create, update and delete a person.
3. Assign and delete card for the specified person.
4. Read all configured entrances.
5. Assign authorization of an entrance to a person/groups.
6. Read any entry and existing transaction of a person.
7. Read all entries and existing transactions filtered by time frame.
8. Direct control of the access to doors.

This document describes the concepts of implementing the Access Professional Edition API for Cardholder Data on the 3rd party application.

3.1 Fundamentals

The API for Cardholder Data consists of methods that are used to manage the objects of the Access Professional Edition database. The API comes in the form of standard libraries, where methods and object classes are operated within C# programming environments.
For C# applications, a set of interop wrappers is provided, namely Bosch_APE_API_CS.dll.

4 Installation

When installing the API for Cardholder Data, the documentation of the interfaces is included along with the application. The document is in the help format.

Supported platforms are:
- Windows 7 SP1 (32-bit, 64-bit in 32-bit emulation mode), .NET framework 3.5 required
- Windows 10 (64-bit), .NET framework 3.5 required

All Microsoft updates and hotfixes are expected to be installed on target PCs. Graphics card drivers are also expected to have the latest officially released version.

For more details please refer to the API for Cardholder Data Installation Packages.

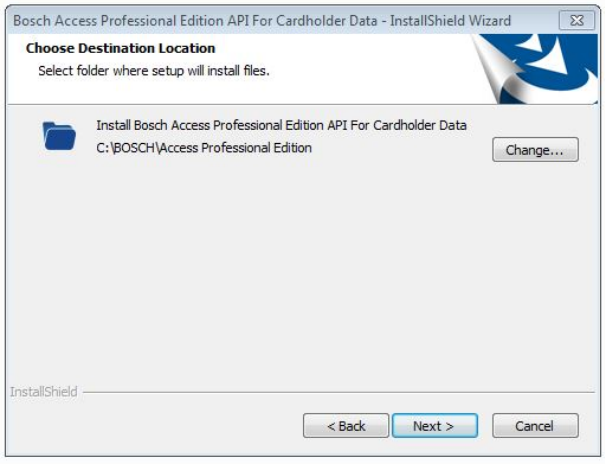
The **Bosch APE_API_CS.DLL** file is used by the developer who is using C# as the development language. This DLL contains several classes, header files and methods. The main class for this DLL is **APE_API** which contains method where the 3rd party applications can be used to perform operations in the Access Professional Edition.

4.1 Installation API for Cardholder Data

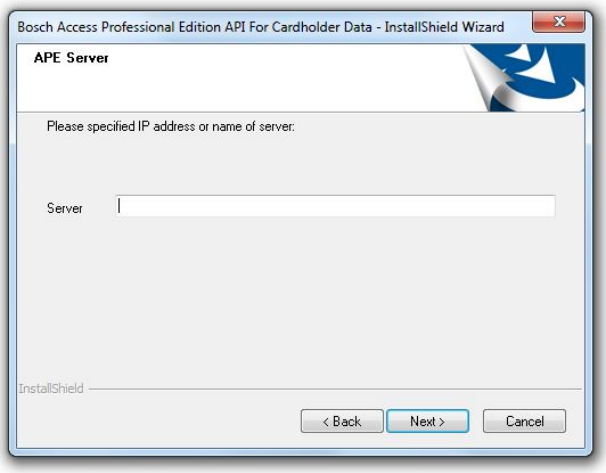
Make sure that you have at least:
- an **APE 3.2 basic license,**
- an **ASL-APE3P-API license**

If these preconditions apply, start the **APE Configurator > Configuration > License activation**. Make sure that the **APE-API For Cardholder Data license activation** is active on your APE system.
The API for cardholder data can be installed :
- On the same PC, where the APE server is installed
- On the same PC, where the APE client is installed
- On a stand-alone PC, no APE client or server is installed on this machine.

Run the **API for cardholder installation** process until it is completed.



If an Access Professional Edition server or client is already installed on the PC, choose the recommended folder on the same drive.
During Installation, you are asked to insert the PC name or IP address of the computer, which hosts the APE server.



For a functional verification of the installation, start the demo app and connect to the **Access Professional Edition Server**.

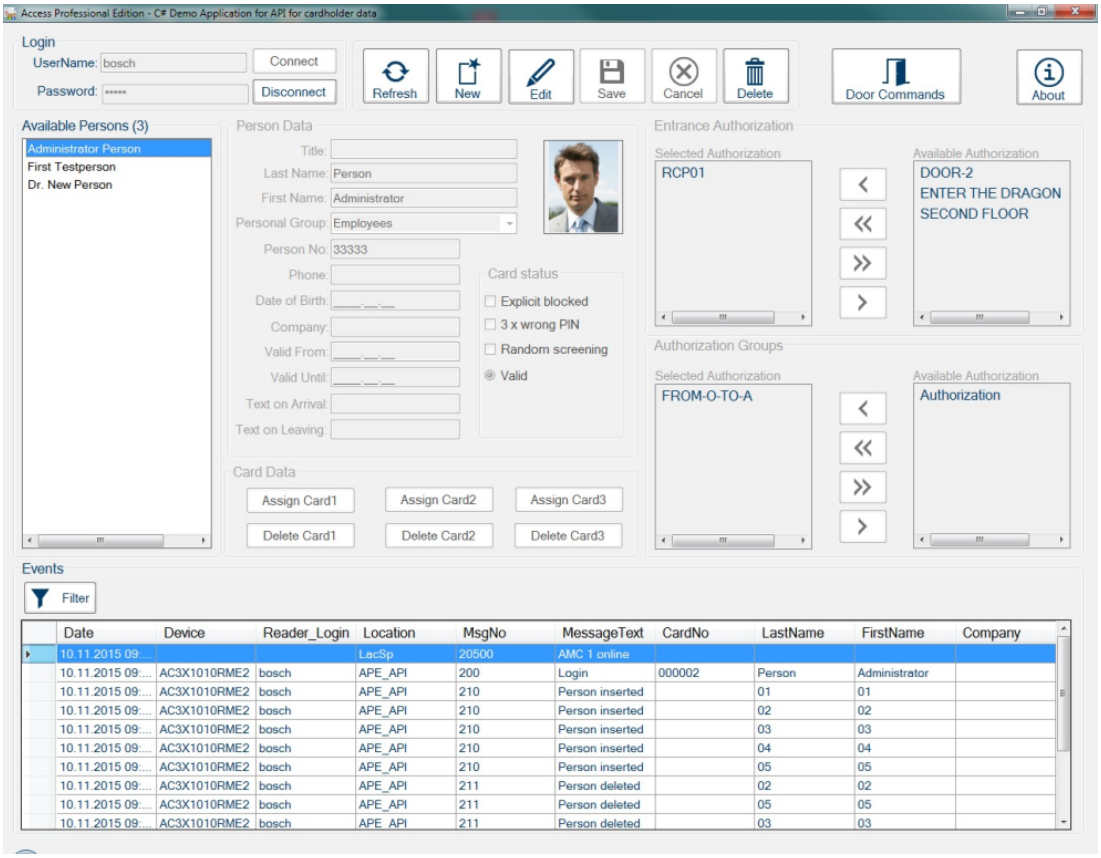
4.2 Version Information

Caution!
API for Cardholder Data and the Access Professional Edition work properly together only as long as both are compatible.
For details refer to the release notes of the Access Professional Edition.

4.3 Demo Application

A demo application is available as an example for the usage of the API for Cardholder Data. The demo application is a simple application with one main screen to demonstrate all available API methods.

Notice!
The Demo Application is intended for demonstration purpose only!
It is not designed to replace the Access Professional Edition Client in a productive environment.



The following transactions can be practiced in the **Demo Application**:

- Create a person in the Access Professional Edition
- Read all the person from the Access Professional Edition
- Read a person particular information
- Delete a person in the Access Professional Edition except administrator user
- Get the available autorizations [entrances]
- Get a new card number which can be used while creating or updating a person
- Get a person's entry and exist transaction
- Handle Open-/Close door commands

5 List of Classes

The following section lists all the classes and their descriptions.

5.1 Person Class

This is the object that holds detailed information of a person.

5.1.1 APE_Person Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
Authorizationlist	Assigned authorization list with authno [integer Array]
Person ID	Person ID [integer]
PersNo	Person Number [integer]
Valid	Person status [integer]
PersonalGroup	PersonGroup ID [integer]
LastName	Person last name [string]
FirstName	Person first name [string]
Title	Person title [string]
DisplayName	Person display name [string]
Name	Person Name [string]
Company	Company information [string]
HandyNo	Person mobile number [string]
DateOfBirth	Date of birth of a person with format yyyymmdd [string]
Pin	Verification pin with 4 digits [string]
TextComing	Text to be displayed in reader on arrival [string]
TextLeaving	Text to be displayed in reader on leaving [string]
PictureFullFileName	Full file path of person picture [string]
PictureFileName	Person picture file name [string]
CardValidFrom	Card valid from with format yyyymmdd [string]
CardValidUntil	Card expiry date with format yyyymmdd [string]
Remarks	Remarks about the person [string]
Memo	Person memo [string]
CardNo1	First card number [integer]
CodeNo1	First card code [string]
CardVersion1	First card version [integer]
CodeNoVersion1	First card code version [integer]

Member Variables	Description
CountryCode1	First card country code [integer]
CustomerCode1	First card customer code [integer]
CardNo2	Second card number [integer]
CodeNo2	Second card code [string]
CardVersion2	Second card version [integer]
CodeNoVersion2	Second card code version [integer]
CountryCode2	Second card country code [integer]
CustomerCode2	Second card customer code [integer]
CardNo3	Third card number [integer]
CodeNo3	Third card code [string]
CardVersion3	Third card version [integer]
CodeNoVersion3	Third card code version [integer]
CountryCode3	Third card country code [integer]
CustomerCode3	Third card customer code [integer]
AreaId	Person location [integer]
AssignedAuthorizationList	List of assigned authorization [List<integer>]
AssignedAuthGroupList	List of assigned authorization group [List<integer>]

5.2 Authorization Class

This class holds the members of authorization.

5.2.1 APE_Authorization Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
AuthNo	Authorization Number [integer]
ValidFrom	Authorization valid from with yyyymmdd format [string]
ValidTo	Authorization valid expiry with yyyymmdd format [string]
AuthName	Authorization name [string]
Type	Authorization type [string]

5.3 AuthorizationGroup Class

This class holds the members of authorization groups.

5.3.1 APE_AuthGroup Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
GroupNo	Authorization group Number [integer]
ValidFrom	Authorization valid from with yyyymmdd format [string]
ValidTo	Authorization valid expiry with yyyymmdd format [string]
AuthName	Authorization group name [string]

5.4 PersonGroup Class

This class holds the details of person group.

5.4.1 APE_PersonGroup Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
PersonGroupID	Person Group ID [integer]
PersonGroupName	Person Group Name [string]
PersonGroupDescription	Person Group Description [string]
Active	Group status [integer]

5.5 Entrances Class

This class holds the details of entrances.

5.5.1 APE_Entrances Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
No	Entrance Number [integer]
Name	Entrance Name [string]

5.6 Message Class

This class holds the details of messages.

5.6.1 APE_Message Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
No	Message Number [integer]
Name	Message Description [string]

5.7 Event Class

This class holds the details of single event.

5.7.1 APE_Event Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
LACNo	LAC Number [integer]
GID	GID used in AC config entrance dialog [integer]
DID	Device ID [integer]
DoorNo	Door number [integer]
DestNo	Destination [integer]
Date	Event date and time [string]
MsgNo	Message number [integer]
CodeNo	Card code for any card related events [string]
ExtraData	Extra event information [string]
Device	Device name in which the event is generated [string]
Reader	Reader name in which the event is generated [string]
Cat	Event category normal event/ alarm[integer]
MessageText	Event description [string]
PersonID	Person ID who’s involved in event [string]
CardNo	Card number who’s involved in event [string]
PersonNo	Person number who’s involved in event [string]
LastName	Person last name who’s involved in event [string]
FirstName	Person first name who’s involved in event [string]
Company	Company ID [string]
MessageType	Event type [string]
AdditionalInfo	Additional information about the event [string]
DateLAC	Date in LAC [string]
AlarmCam	Alarm camera ID [integer]
CamImage	Surveillance camera ID [integer]

5.8 EventFilter Class

This class holds the details of events filter.

5.8.1 APE_EventFilter Class

This object is used by the developer who is using C# as the development language.

Member Variables	Description
LastName	Lastname of person [integer]
Company	Company name [string]
Messages	Message ID, more than one message can be assigned using ‘;’ separated [string]
Entrances	Entrance ID, more than one entrance can be assigned using ‘;’ separated [string]
CardNo	Card number [string]
FromDate	From date [string]
ToDate	To date [string]

6 List of Methods

The following section lists all the methods that are applicable for the classes.

6.1 Connect()

Description:
Use this method to create an instance to this class and get connected to the Access Professional Edition server.

Parameters:

C#	`Username`	Valid APE username.
	`Password`	Valid APE password.

Definition:

C#	`int Connect(String^ Username,String^ Password);`

Example:

C#	`APE_API ac = new APE_API();` `int ret = ac.Connect("abc","xyz");`

6.2 Disconnect()

Description:
Use this method to dispose objects and disconnect from Access Professional Edition. This method must be called while closing the application or during disconnect.

Parameters:
None.

Definition:

C#	`int Disconnect();`

Example:

C#	`APE_API ac = new APE_API();` `m_ac.Disconnect();`

6.3 ReadAllPerson()

Description:
Use this method to get all available person information. The person list will be filled on the parameter.

Parameters:

C#	`APE_Person object list`

Definition:

C#	`int ReadAllPerson(List<APE_Person^>^ m_AcPerlist);`

Example:

C#	`List<APE_Person> personlist = new List<APE_Person>();` `int val = ac.ReadAllPerson(personlist);`

6.4 GetPerson()

Description:
Use this method to get detailed information of a specific person.

Parameters:

C#	`PersonID`	Required Person ID
	`APE_Person object`	This method fills the person details.

Definition:

C#	`int GetPerson(int PersonID, APE_Person^ AcPerson);`

Example:

C#	`APE_Person person = new APE_Person();` `int ret = ac.GetPerson(1, person);`

6.5 CreatePerson()

Description:
Use this method to create a person with specified information. Before calling this method, all required information has to be filled in the Person object.

Parameters:

C#	`APE_Person object`	Person object with details.

Definition:

C#	`int CreatePerson(APE_Person^ APEPerson);`

Example:

C#	APE_Person p;
	p.title = "Mr";
	p.lastName = "Person";
	p.firstName ="Test";
	p.personalGroup = 1;
	int ret = ac.CreatePerson(p);

6.6 UpdatePerson()

Description:
Use this method to update a person with specified information. Before calling this method, any modified information has to be filled in the existing Person object.

Parameters:

C#	APE_Person object	Person object with details.

Definition:

C#	int UpdatePerson(APE_Person AcPerson);

Example:

C#	APE_Person pers;
	pers.title = "Mr";
	pers.lastName = "Person1";
	pers.firstName ="Test1";
	pers.personalGroup = 1;
	int ret = ac.UpdatePerson(pers);

6.7 DeletePerson()

Description:
Use this method to delete a specified person.

Parameters:

C#	APE PersonID	Person ID to be deleted.

Definition:

C#	int DeletePerson(int PersonID);
	or
	int DeletePerson(APE_Person^ APEPerson);

Example:

C#	`int ret = ac.DeletePerson(PersonID);`
	or
	`int ret = ac.DeletePerson(m_APE_PersonObj);`

6.8 GetPersonGroupList()

Description:
Use this method to get the person group list.

Parameters:

C#	`APE_Person object list`	This method fills the list of available person groups.

Definition:

C#	`int GetPersonGroupList(List<APE_PersonGroup^>^ m_AcPerGrouplist);`

Example:

C#	`List<APE_PersonGroup>`
	`AcPersonGroupList = new List<APE_PersonGroup>();`
	`int ret = ac.GetPersonGroupList(AcPersonGroupList);`

6.9 GetAuthorizationList()

Description:
Use this method to get the available authorizations [Entrance] list.

Parameters:

C#	`APE_Authorization object list`	This method fills the list of available authorizations.

Definition:

C#	`int GetAuthorizationList(List<APE_Authorization^>` `^m_AcPersonAuthorizationList);`

Example:

C#	`List<APE_Authorization>`
	`AcPersonAuthList new List<APE_ Authorization >();;`
	`int ret = ac.GetAuthorizationList(AcPersonAuthList);`

6.10 GetAuthGroupList()

Description:
Use this method to get the available authorizations [Entrance] group list..

Parameters:

C#	APE_AuthGroup object list	This method fills the list of available authorization groups.

Definition:

C#	int GetAuthGroupList(List<APE_AuthGroup^>^m_AcPerAuthGroupList);

Example:

C#	List<APE_AuthGroup> AcPerAuthGroupList = new List<APE_ AuthGroup >();; int ret = ac.GetAuthGroupList(AcPerAuthGroupList);

6.11 GetEntranceList()

Description:
Use this method to get the available entrance list. It can be used while defining event filter with entrances.

Parameters:

C#	APE_Authorization object list	This method fills the list of available authorizations.

Definition:

C#	int GetEntrancesList(List<APE_Entrance^>^ m_AcEntrancelist);

Example:

C#	List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>(); int val = ac.GetEntrancesList(availableEntrancelist);

6.12 OpenEntranceOnce()

Description:
Use this method to open a specific Entrance.
(Select the Entrance from the available Entrance List.)

Parameters:

C#	APE_Entrance object. **This method will open the entrance.**

Definition:

C#	`int OpenEntranceOnce (APE_Entrance openEntrance);`

Example:

C#	`List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>();`
	`int ret = ac.GetEntrancesList (availableEntranceList);`
	`if ( ret == 1 && availableEntranceList.Count > 0 )`
	`{`
	`int iSerachIndex = 0 ; // use 0 for example`
	`APE_Entrance openEntrance`
	`= availableEntranceList[iSerachIndex];`
	`ret = ac.OpenEntranceOnce (openEntrance);`
	`}`

6.13 OpenEntrancePermanent()

Description:
Use this method to Open a specific Entrance permanently.
(Select the Entrance from the available Entrance List)

Parameters:

C#	APE_Entrance object. **This method will open the entrance permanently**

Definition:

C#	`i int OpenEntrancePermanent (APE_Entrance openEntrance);`

Example:

C#	`List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>();`
	`int ret = ac.GetEntrancesList (availableEntranceList);`
	`if ( ret == 1 && availableEntranceList.Count > 0 )`
	`{`
	`int iSerachIndex = 0 ; // use 0 for example`
	`APE_Entrance openEntrance`
	`= availableEntranceList[iSerachIndex];`
	`ret = ac.OpenEntrancePermanent (openEntrance);`
	`}`

6.14 CloseEntrancePermanent()

Description:
Use this method to Close a specific Entrance.
(Select the Entrance from the available Entrance List.)

Parameters:

C#	APE_Entrance object. **This method will close the entrance.**

Definition:

C#	`int OpenEntrancePermanent (APE_Entrance openEntrance);`

Example:

C#	`List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>();`
	`int ret = ac.GetEntrancesList (availableEntranceList);`
	`if ( ret == 1 && availableEntranceList.Count > 0 )`
	`{`
	`  int iSerachIndex = 0 ; // use 0 for example`
	`  APE_Entrance openEntrance`
	`=  availableEntranceList[iSerachIndex];`

6.15 LockEntrance()

Description:
Use this method to Lock a specific Entrance.
(Select the Entrance from the available Entrance List.)

Parameters:

C#	APE_Entrance object. **This method will lock the entrance.**

Definition:

C#	`int LockEntrance (APE_Entrance openEntrance);`

Example:

C#

```
List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>();
int ret = ac.GetEntrancesList (availableEntranceList);
if ( ret == 1 && availableEntranceList.Count > 0 )
{
    int iSerachIndex = 0 ; // use 0 for example
    APE_Entrance openEntrance
=    availableEntranceList[iSerachIndex];
    ret = ac.LockEntrance (openEntrance);
}
```

6.16 UnlockEntrance()

Description:
Use this method to Unlock a specific Entrance.
(Select the Entrance from the available Entrance List.)

Parameters:

C#	APE_Entrance object. **This method will unlock the entrance.**

Definition:

C#

```
int UnlockEntrance (APE_Entrance openEntrance);
```

Example:

C#

```
List<APE_Entrance> availableEntrancelist = new List<APE_Entrance>();
int ret = ac.GetEntrancesList (availableEntranceList);
if ( ret == 1 && availableEntranceList.Count > 0 )
{
    int iSerachIndex = 0 ; // use 0 for example
    APE_Entrance openEntrance
=    availableEntranceList[iSerachIndex];
    ret = ac.UnlockEntrance (openEntrance);
}
```

6.17 GetMessageList()

Description:
Use this method to get the available message [event] list. It can be used while defining event filter with message.

Parameters:

C#	APE_Message object list	This method fills the list of available messages [event].

Definition:

C#	`int GetMessagesList(List<APE_Message^>^ m_AcMessagelist);`

Example:

C#	`List<APE_Message> messagelist = new List<APE_Message>();` `int val = ac.GetMessagesList(messagelist);`

6.18 GetNewCardNumber()

Description:
Use this method to get a new card number which does not conflict with other numbers.

Parameters:

C#	`PersonGroup`	Person group ID needed to generate available card number.

Definition:

C#	`int GetNewCardNumber(int personGroup);`

Example:

C#	`int cardNumber = ac.GetNewCardNumber(1);`

6.19 GetPersonCount()

Description:
Use this method to get the count of an available person.

Parameters:
None

Definition:

C#	`int GetPersonCount();`

Example:

C#	`int count = ac.GetPersonCount();`

6.20 ExecuteEventFilter()

Description:
Use this method to apply filter condition to the runtime events as well as to obtain the history.

Notice!
In order to avoid duplication, please clear the local filter store as the new list will be obtained which contains filtered events. For example:
After applying filter like, today's events only and also event generated by reader1: will return a new list of events occurred today on reader1 and also any real-time event which satisfies this filter. And on removing the filter, the API provides the list of current date events with no filter and followed by the real-time events.

Parameters:

C#	`APE_EventFilter object`	This parameter hold the object holds the filter definition which will be applied. Apply default values to this object if the user wants to remove the filter.

Definition:

C#	`int ExecuteEventFilter(APE_EventFilter^ Filter);`

Example:

C#	`APE_EventFilter filter = new APE_EventFilter();`
	`filter.FromDate = ”2014/12/28”;`
	`filter.ToDate = ”2014/12/28”;`
	`filter.LastName = “TestUser”;`
	`filter.Company = “Bosch”;`
	`filter.CardNo = “456”;`
	`filter.Entrances = “Door1;Door2”;`
	`filter.Messages = “1;2;3;4”;`
	`ac.ExecuteEventFilter(filter);`

Default:
Filter is applied for current date only.

6.21 ReceiveEvent()

Description:
Use this method to register external application to receive the events.

Parameters:
None

Definition:

C#	`delegate void GetEventHandler(Object^ sender,APEEventArgs^ e);` `event GetEventHandler^ RecieveEvent;`

Example:

C#	`ac.RecieveEvent += new APE_API.GetEventHandler(DisplayEvent);`
	`private void DisplayEvent(Object sender,APEEventArgs e)`
	`{`
	`Event m_newEvent = e.evt;`
	`}`

6.22 GetVersion()

Description:
Use this method to get the current APE_API internal version information.

Parameters:

C#	major minor build revision	pointer to an int variable that receives major version number pointer to an int variable that receives minor version number pointer to an int variable that receives build number pointer to an int variable that receives revision number

Example:

C#	int major = 0;
	int minor = 0
	int build = 0;
	int revision = 0;
	ac.GetVersion(ref major, ref minor, ref build, ref revision);

6.23 GetExpectedVersion()

Description:
Use this method to get the expected APE compatibility version information.

Example:

C#	`string sExpVer = ac.GetExpectedVersion();`

6.24 GetServerVersion()

Description:
Use this method to get the the real APE compatibility version information.

Example:

C#	`string sSrvVer = ac.GetServerVersion();`

6.25 RefreshData()

Description:
Use this method to refresh all data (person group list, authorization list, authorization group list, entrance list, message list, etc.) used by the API, from APE server.

Parameters:
None

Definition:

C#	int RefreshData();

Example:

C#	int rc = ac.RefreshData();

Notice!
It is recommended to call this method before calling any of the "Get" methods used to get different data (person group list, authorization list,authorization group list, entrance list, message list, etc.).
For performance reasons, all data (excepting personal data) is cached by the API and any changes made on the APE server side will be first available to the API after refreshing the data.

7 List of Return Values

API_ReturnValue is an enum member variable which contains all the return error codes. The list of return error codes are described in the table below.

Error Code	Description
0	Operation failed
1	Operation succeeded
2	Unable to delete the current user
3	Not authorized to perform the operation
4	Failed to read the configuration file
5	Failed to read the custom data file
6	Error occurred during the Database initialization
7	Error while opening the Database
8	Failed to open a table
9	License tampered
10	Unable to get the person user rights
11	Error while obtaining configuration data
12	Error during reading areas
13	Error during reading all person
14	Error accessing system number table
15	Maximum person count exceeded
16	Error on getting person ID
17	Customer code invalid
18	Invalid country code
19	Invalid card code
20	Invalid card version
21	The specified card code conflicts with his/her existing card code
22	The specified card code conflicts with other person card code
23	The specified card number conflicts with other person card number
24	The specified card number conflicts with his/her existing card number
25	Failed to obtain a new card number
26	Selected person group is invalid
27	Unable to read the Log file
28	API license is not activated
29	Cannot delete administrator user

Error Code	Description
30	Demo license expired
31	Codeno/Cardno combination error
32	Login rejected. Too many clients
33	Login rejected. Conflicting compatibility versions
34	General error reading license
35	License details error

Bosch Sicherheitssysteme GmbH
Robert-Bosch-Ring 5
85630 Grasbrunn
Germany
www.boschsecurity.com