UNION COMMUNITY
Server SDK For Windows

Programmer's Guide
Version 4.0

October, 2009

Software Development Department

UNION COMMUNITY CO., LTd.

Union Community Co., LTD.
Hyundai Topics Bldg., 44-3, Bangi-dong, Songpa-gu, Seoul 138-050 Korea
Tel : +82-6488-3000, Fax : +82-6488-3099 http://www.unioncomm.co.kr

USER License Agreement for Software Developer's Kit

Designed by Union Community Co., Ltd

This agreement is a legal usage license agreement between Union Community Co., Ltd. and the user. If you do not agree with the terms and condition of the agreement, please return the product promptly. If you return the product, you will receive a refund.

1. Usage License

UNION COMMUNITY Co., Ltd. Grants licensee to use this SDK a personal, Limited, non-transferable, non-exclusive right to install and use one copy of the SDK on a single computer exclusively.
The software is considered 'being used' if it is stored in a computer's main or other storage device. The number of software copies will be determined by taking the greater number of the number of computers 'used' by the software and the number of computers with the software stored.
Licensee may use the SDK solely for developing, designing, and testing UNION software applications for use with UNION products ("Applications").

2. Right to Upgrade

If you have purchased the software by upgrading an older version, the usage license of the old version is transferred to the new version. However, you may only use the old version under the condition that the old and new versions are not running simultaneously. Therefore, you are prohibited from transferring, renting or selling the old version. You maintain the usage license for the program and ancillary files that are in the old version but not in the new version.

3. Assignment of License

If you wish to transfer the usage license of this software to a third party, you must first obtain a written statement indicating that the recipient agrees with this agreement. You must then transfer the original disk and all other program components, and all copies of the program must be destroyed. After the transfer is complete, you must notify UNION COMMUNITY Co., Ltd. to update the customer registration.
Licensee shall not rent, lease, sell or lend the software application developed using the SDK to a third party without UNION's prior written consent.

Licensee shall not copy and redistribute the SDK without UNION's prior written consent.
No other uses and/or distribution of the SDK or Sample Code are permitted without UNION's prior written consent. UNION reserves all rights not expressly granted to Licensee.

4. Copyright

All copyrights and intellectual properties of the software and its components belong to UNION COMMUNITY Co., Ltd. and these rights are protected under Korean and international copyright laws. Therefore, you may not make copies of the software other than for your backup purposes. In addition, you may not modify the software other than for reverse-engineering purposes to secure compatibility. Finally, you may not modify, transform or copy any part of the documentation without written permission from UNION COMMUNITY. (If you're using a network product, you may copy the documentation in the amount of the number of users)

5. Installation

An individual user can install this software in his/her PCs at home and office, as well as in a mobile PC. However, the software must not be running from two computers simultaneously. A single product can be installed in two or more computers in one location, but one of those computers must have a usage rate of at least 70%. If another computer has a usage rate of 31% or higher, another copy of the software must be purchased.

6. Limitation of Warranty

UNION COMMUNITY Co., Ltd. guarantees that the CD-ROM and all components are free of physical damage for a year after purchase.
UNION DISCLAIMS ALL WARRANTIES NOT EXPRESSLY PROVIDED IN THIS AGREEMENT INCLUDING, WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE. If you find any manufacture defect within the warranty period, we will replace the product. You must be able to prove that the product has been purchased within a year to receive a replacement, but we will not replace a product damaged due to your mishandling or negligence. UNION COMMUNITY Co., Ltd. does not guarantee that the software and its features will satisfy your specific needs, and is not liable for any consequential damages arising out of the use of this product.

7. Liabilities

UNION COMMUNITY Co., Ltd. is not liable for any verbal, written or other agreements made by third parties, including product suppliers and dealers.

8. Termination

This agreement is valid until the date of termination. However, the agreement shall terminate automatically if you damage the program or its components, or fail to comply with the terms described in this agreement.

9. Customer Service

UNION COMMUNITY Co., Ltd. makes every effort to provide registered customers with technical assistance and solutions to problems regarding software applications under certain system environments. When a customer submits a suggestion about any inconvenience or anomaly experienced during product usage, UNION COMMUNITY Co., Ltd. will take corrective action and notify the customer of the result.

10. General Terms

You acknowledge that you have read, understood and agree with the terms of this agreement. You also recognize the fact that this agreement has precedence over user agreements of older versions, past order agreements, advertisement notifications and/or other written agreements.

11. Contact

If you have any questions about this agreement, please contact UNION COMMUNITY Co., Ltd. via telephone, fax or e-mail.

목 차

1. Overview

UCS(UNION COMMUNITY Server) SDK는 ㈜유니온커뮤니티의 네트워크형 지문인식 단말기와 연동 가능한 응용 프로그램 개발을 쉽게 할 수 있도록 High Level SDK 형태로 제작 되었다.

UCS SDK는 지문인식 서버 응용 프로그램 개발에 필요한 인터페이스(Application Programming Interface, API)를 제공하기 위한 것으로 지문 등록 및 인증을 위해 UCBioBSP SDK 와 함께 사용 할 수 있다.

1.1 적용

UCS SDK는 ㈜유니온커뮤니티에서 제공하는 네트워크형 단말기 제품과 연동 할 수 있는 서버 응용 프로그램 인터페이스(Server Application Programming Interface)를 정의 하고 있다. 따라서 출입통제, 근태, 식수, 학사 관리등의 응용프로그램 개발 시에 적용하여, 보다 쉽고, 안정적인 프로그램을 개발하는데 활용 할 수 있다.

1.2 특징

■ 중앙 집중 관리 방식
단말기가 UCS SDK에 접속 하는 방식으로 중앙에서 모든 단말기들을 집중 관리 할 수 있으며, 이러한 방식은 공중망을 이용한 네트워크 구성에서 많은 이점을 가지고 있다.
만일 종량제 (회선을 사용한 시간만큼 이용 요금을 매기는 제도)를 사용하는 공중망을 이용하는 경우 사용하는 SDK의 서버 기능을 실행하지 않고 단말기로 직접 접속하여 관리하는 방법을 사용 할 수 있다.

■ 단말기 관리를 위한 다양한 API 제공
사용자 관리, 로그 관리, 출입통제 관리등을 위한 다양한 API를 제공 한다.

■ 다양한 개발 모듈과 샘플 소스 제공
UCS SDK에서는 C/C++ 개발자 뿐만 아니라 Visual Basic 이나 Delph, DotNet 등을 사용 하는 개

발자를 위하여 이들 툴에서 개발을 쉽게 할 수 있도록 COM 기반의 모듈과 DotNet Class 라이브러리를 함께 제공한다. 또한 SDK 사용에 필요한 샘플 소스를 함께 제공한다.

■ 다양한 인증 방식 제공

사용자 식별을 위한 수단으로 지문 이외에 비밀번호, 카드와 이들에 대한 다양한 조합의 인증 방식을 제공 한다.

1.3 개발 환경

UCS SDK에서 제공하는 모든 모듈은 VC++ 6.0에서 컴파일 되어졌으며, Visual C++ 등의 대부분의 32bit 컴파일러에서는 이 SDK를 사용하여 프로그래밍이 가능하다.

또한 C/C++ 개발자 뿐만 아니라 Visual Basic 이나 Delphi, DotNet등을 사용 하는 개발자를 위하여 이들 툴에서 개발을 쉽게 할 수 있도록 COM 기반의 모듈과 DotNet 클래스 라이브러리를 함께 제공한다. 모듈 사용 방법에 대해서는 각각의 샘플 소스를 참조 한다면 많은 도움이 될 것이다.

1.4 모듈 구성

■ **Basic 모듈 : UCSAPI40.dll**

UCS SDK의 핵심 모듈로서 단말기와의 통신과 관련되 모든 기능을 구현하고 있는 메인 모듈이라 할 수 있다. UCS SDK를 이용해 개발을 할 경우 이 모듈은 반드시 포함되어야 한다.

C, C++에서 사용 할 수 있는 API들을 제공하고 있다.

관련된 Sampe 코드는 제공되는 Samples 폴더의 DLL 폴더에서 찾을 수 있다.

■ **COM 모듈 : UCSAPICOM.dll**

Visual Basic 이나 Delphi등의 RAD(Rapid Application Development) Tool 사용자를 지원하기 위한 목적으로 개발된 COM(Component Object Model) 모듈이다.

UCSAPICOM.dll은 UCSAPI40.dll 보다 상위레벨에 존재하기 때문에 UCSAPI40.dll이 반드시 같이 존재해야 동작 할 수 있다. 또한 UCSAPI40.dll에서 제공하는 모든 기능을 제공하지는 않지만 반대로 UCSAPI40.dll에서 제공하지 않는 기능도 일부 가지고 있다.

관련된 Sampe 코드는 제공되는 Samples 폴더의 COM 폴더에서 찾을 수 있다.

■ **DotNET 모듈 : UNIONCOMM.SDK.UCSAPI40.dll**

Microsoft의 .NET 환경에서 사용되는 C#, VisualBasic.NET 등의 언어에서 사용 가능한 .NET Class library 모듈이다. COM 모듈과 마찬가지로 UNIONCOMM.SDK.UCSAPI40.dll 은 UCSAPI40.dll 보다 상위레벨에 존재하기 때문에 UCSAPI40.dll이 반드시 같이 존재해야 동작 할 수 있다.

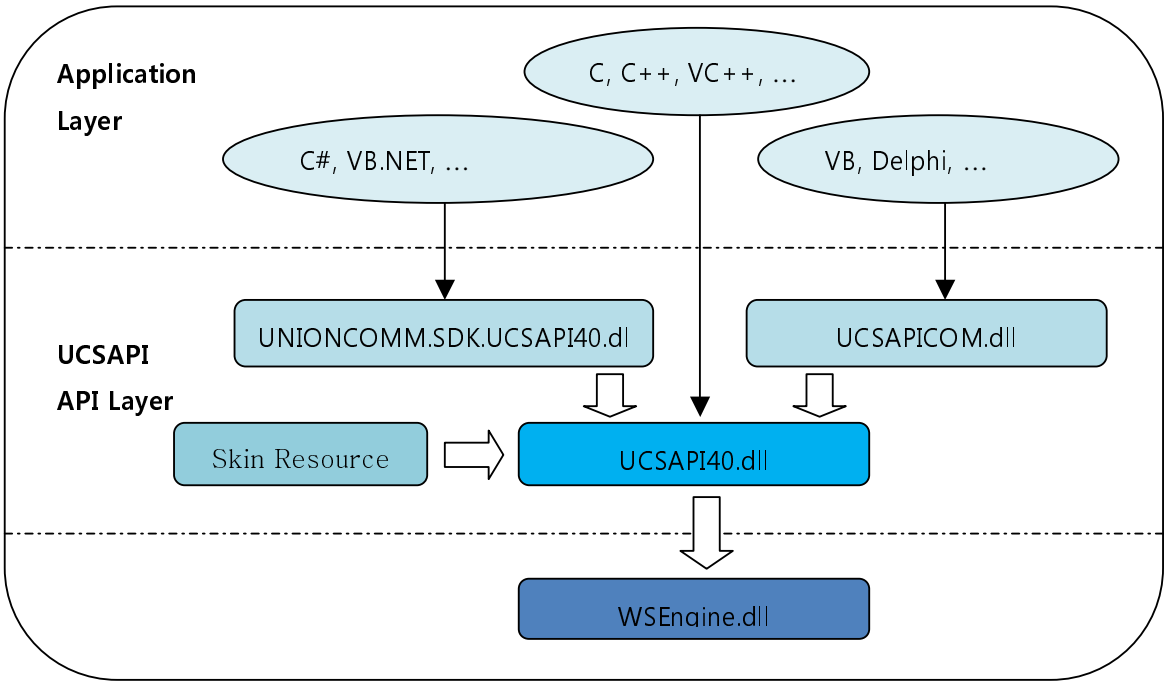
관련된 Sample 코드는 제공되는 Samples 폴더의 dotNET 폴더에서 찾을 수 있다.

■ **Winsock Engine 모듈 : WSEngine.dll**

Socket IO(Input/Output)를 담당하는 모듈이다. WSEngine.dll은 UCSAPI40.dll 보다 하위레벨에 존재한다.

1.5 개발 모델

개발 모델에 대한 구조는 아래와 같다.



1.5 용어 설명

■ **터미널 (Terminal)**

㈜유니온커뮤니티에서 제공하는 네트워크형 지문인식 단말기를 말한다.

■ **클라이언트 (Client)**

제공하는 네트워크형 지문인식 단말기와 통신하고자 하는 응용 프로그램을 말한다.

■ **클라이언트 ID (ClientID)**

클라이언트 ID는 클라이언트/서버 모델로 응용 프로그램을 개발하고자 할 경우 사용 된다.
서버 모듈은 멀티 클라이언트 환경을 지원하기 위해 각각의 클라이언트들을 구분하기 위한 Key 가 필요하며 이를 클라이언트 ID라 말한다.

■ **1:1 인증 (1 to 1, Verification)**

개인의 신원을 확인하기 위해서 사용자 ID에 해당하는 지문 템플릿(또는 카드, 비밀번호)과 제출된 샘플을 비교하는 일대일 의 처리 과정이다.

■ **1:N 인증 (1 to N, Identifacion)**

개인의 신원을 확인하기 위하여 일부 혹은 모든 지문 템플릿 과 제출된 샘플을 비교하는 일대다 수의 처리 과정이다.

■ **인증 타입 (Authentication Type)**

사용자 인증 시 사용되는 인증 타입을 정의 한다.
사용자가 인증을 위해 UserID또는 UniqueID를 단말기로부터 입력하고 지문인식 단말기가 서버 인증 방식을 사용하는 경우 단말기는 서버에 등록된 사용자의 인증 타입을 획득하기 위하여 서버로 인증 타입을 요청 하게된다.

Type	Value	Contents
1:N Fingerprint	0	1:N Fingerprint Authentication
1:1 Fingerprint	1	1:1 Fingerprint Authentication
Card & Fingerprint	2	Smart Card에 Fingerprint 정보를 저장하여 입력 Fingerprint과 저장 Fingerprint간 1:1 인증을 수행

Card	3	Card Authentication
Password	4	Password Authentication

■ **등록 인증 타입 (Registration Authentication Type)**

사용자 등록시 가 가질 수 있는 인증 타입을 정의 한다.

사용자 식별을 위한 수단으로 지문 이외에 비밀번호, 카드와 이들에 대한 다양한 조합의 인증 방식을 제공 한다.

Type	Contents
Fingerprint	인증 수단으로 지문을 사용 한다.
Fingerprint Card	인증 수단으로 지문 카드를 사용 한다. 지문카드는 스마트 카드에 개인의 지문 템플릿을 저장하여 인증 시 입력된 샘플과 저장 템프릿간 1:1 인증을 수행한다.
Password	인증 수단으로 비밀번호를 사용 한다. 비밀번호는 최대 8자리의 문자열 값이다.
Card	인증 수단으로 카드를 사용 한다.
Card or Fingerprint	인증 수단으로 카드 또는 지문을 사용 한다.
Card and Fingerprint	인증 수단으로 카드와 지문의 조합을 사용 한다.
Card or Password	인증 수단으로 카드 또는 비밀번호를 사용 한다.
Card and Password	인증 수단으로 카드와 지문의 조합을 사용한다.
(ID and Fingerprint) or (Card and Fingerprint)	인증 수단으로 ID와 지문의 조합 또는 카드와 지문의 조합을 사용한다.
(ID and Password) or (Card and Password)	인증 수단으로 ID와 비밀번호의 조합 또는 카드와 비밀번호의 조합을 사용한다.
Fingerprint and Password	인증 수단으로 지문과 비밀번호의 조합을 사용한다.
Fingerprint or Passwrod	인증 수단으로 지문 인증 실패 후 비밀번호를 사용한다.
Card and Password and Fingerprint	인증 수단으로 카드와 비밀번호, 지문의 조합을 사용한다.

■ **지문 인증 레벨 (Security Level)**

지문 인증 시 사용할 보안 수준을 정의 한다, 그 범위는 1 ~ 9의 값을 갖는다. UCS SDK는 개발자들에게 다음과 같은 레벨 값을 사용하도록 권장 하고 있다. 레벨 값이 높을수록 본인 거부 율(FRR)이 높아지며, 타인 인증 율(FAR)은 낮아진다. UCS SDK는 Verification(1:1) 인증 시 레벨 4와 Identification(1:N) 인증 시 레벨 5를 기본 레벨로 권장한다.어플리케이션은 기본 레벨을 기준으로 FAR이 높은 경우 인증레벨을 상향 조정할 수 있으며, FRR이 높을수록 인증레벨을 하향 조정 할 수

있다.

■ **단말기 인증 방식 (Terminal Authentication Mode)**

사용자의 인증 및 로그 기록에 대한 동작 방식을 정의 한다.

Mode	Value	Content
N / S	0	On-line시 서버에서 사용자 인증을 수행 하며, Off-line 시 단말기에서 사용자 인증을 수행한다. 서버 인증 동안은 서버에서 인증 기록을 저장하며, 네트웍 단절 시에는 단말기에서 인증 기록을 저장하고 서버 연결 시 단말기에 저장된 인증 기록을 서버로 전송한다.
S / N	1	On-line시 단말기에서 사용자 인증을 수행한다. 단 단말기에 저장되어 있지 않은 사용자 인증 요구에 대해서는 서버로 인증 요청 한다. 인증기록은 항상 단말기에 저장하여, On-line시에는 인증 결과 만을 서버로 전송하여 저장 하게 한다.
N / O	2	서버에서만 사용자 인증을 수행 한다. Off-line시에는 사용자 인증을 수행할 수 없다.
S / O	3	단말기에서만 사용자 인증을 수행 한다. On-line시에는 인증 로그만 서버로 전송한다.
S / S	4	단말기는 서버로 접속 시도를 하지 않으며 단지 응용 프로그램의 접속을 대기 한다. 단말기에선 인증 수행을 한다.

■ **단말기 프로그램 모드 (Terminal Application Mode)**

단말기에서 지원하는 프로그램 운영 방식을 정의 한다.

Mode	Value	Content
출입통제	0	단말기를 출입 통제 목적으로 운영 한다.
근태	1	단말기를 근태 관리 목적으로 운영 한다.
식수	2	단말기를 식수 관리 목적으로 운영 한다.

■ **사용자 인증 모드 (User Authentication Mode)**

사용자 인증 시 인증에 대한 목적을 정의 한다. 기본 인증 모드 외에 확장된 모드를 사용하고자 한다면 단말기의 확장 모드 옵션을 설정하여 사용 할 수 있다. 인증 모드는 프로그램을 근태 및 식수 목적으로 운영 할 경우 사용 될수 있다.

Mode	Value	Content
출근	0	출근 모드로 인증 한다.
퇴근	1	퇴근 모드로 인증 한다.

일반	2	일반 모드로 인증 한다.
외근	3	외근 모드로 인증 한다.
복귀	4	복귀 모드로 인증 한다.

■ 로그 얻기 타입

UCS SDK는 단말기로부터 로그 데이터를 얻어 오기 위하여 세가지 타입의 로그 종류를 정의 한다.
단말기의 모델에 따라 내부에 저장 할 수 있는 로그의 개수는 서로 차이가 있으며 최대 저장 용량을 초과시 이전의 기록을 일부 삭제하여 새로운 로그를 저장 하게 된다.

Type	Value	Content
New Log	0	서버로 전송 되지 않은 새로운 로그
Old Log	1	서버로 이미 전송 완료된 로그
All Log	2	단말기에 저장되어 있는 모든 로그

■ 사용자 속성 (User Property)

사용자의 관리자 여부 및 인증 타입을 정의하는 1 바이트 크기의 데이터 필드 이다.
각 Field는 1 or 0의 값을 지정 할 수 있다. 해당 Field의 속성 값을 사용하려면 1을 지정한다.

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	Operation (AND) or (OR)	Card ID	Card	Password	Reserved	Fingerprint

-Admin : 사용자를 단말기 관리자로 지정 할 수 있다.
-Identify : 사용자를 1:N 지문 인증 가능 하도록 지정 할 수 있다.
-Operation : 각각의 인증 타입을 AND or OR 조합으로 묶어 사용 하도록 지정 할 수 있다
여기서 AND 조합인 경우 1, OR 조합의 경우는 0을 설정 한다.
-CardID : RFID를 UserID 또는 UniqueID 처럼 사용하도록 지정 할 수 있다. CardID는 카드의 RFID를 인증 수단으로 사용하지 않고, 단지 UserID 처럼 식별자로 사용하는 것을 말한다.
반드시 다른 인증 타입과 함께 AND 조합으로 지정 되어야 한다.
-Card : 사용자를 Card 인증 가능 하도록 지정 할 수 있다.
-Password : 사용자를 Password 인증 가능 하도록 지정 할 수 있다.
-Fingerprint : 사용자를 지문 인증 가능 하도록 지정 할 수 있다.

사용자 속성은 다음의 12가지 값으로 표현 되어 질 수 있다.

① Fingerprint

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	0	0	0	1

② Fingerprint-Card

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	0	0	1	0

③ Password

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	0	1	0	0

④ Card

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	1	0	0	0

⑤ Card or Fingerprint

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	1	0	0	1

⑥ Card and Fingerprint

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	1	0	1	0	0	1

⑦ Card or Password

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	1	1	0	0

⑧ Card and Password

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	1	0	1	1	0	0

⑨ (ID and Fingerprint) or (Card and Fingerprint)

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit

Admin	Identify	0	1	0	0	0	1

⑩ (ID and Password) or (Card ande Password)

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	1	0	1	0	0

⑪ Fingerprintf and Password

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	1	0	0	1	0	1

⑫ Fingerprintf or Passowrd : 지문 인증 실패 시 패스워드 인증

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	0	0	0	1	0	1

⑬ Card and Password and Fingerprintf

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	0	1	0	1	1	0	1

■ 단말기 관리자 (Terminal Admin)

단말기 관리자는 단말기의 설정 정보를 변경 하거나 사용자를 등록/삭제 할 수 있는 권한을 가진 사용자를 말한다. 단말기에 1명 이상의 단말기 관리자가 저장되어 있는 경우 단말기의 설정 화면으로 진입 시 관리자 로그인 과정이 필요하게 된다. 단말기의 안전한 사용을 위하여 단말기 관리자를 반드시 등록하여 사용기를 권장한다.

■ 단말기 상태

단말기는 자신의 상태 및 단말기에 연결된 장치의 상태 정보를 주기적 또는 상태 변화가 있을 경우 즉시 서버로 상태 정보를 전송 한다.

단말기 잠금 상태:

이 값은 단말기의 조작 가능 상태 값을 난타낸다.

SDK는 단말기의 조작 상태를 Lock/UnLock 상태로 설정 할 수 있으며, Lock 상태에서는 단말기의로 그온과 네트워크의 연결은 유지하지만 관리자를 제외한 단말기로의 조작 및 출입을 불가능하게 한다. Lock 상태로 진입한 단말기는 Key-pad 조작 조차 불가능 하게 된다.

잠금장치 제어 상태:

이 값은 단말기에 연결된 잠금 장치의 상태 값을 나타낸다

SDK는 단말기에 연결 되어있는 잠금 장치의 상태를 열림 상태로 설정/해제 할 수 있으며, 열림 상태에서는 사용자의 인증 없이 출입이 가능하게 된다.

잠금장치 모니터링 상태:

이 값은 모니터링이 지원되는 잠금 장치로부터 수신되는 장금장치의 열림/닫힘 상태 값을 난타낸다

단말기 뚜껑 상태:

이 값은 단말기 뚜껑의 열림/닫힘 상태 값을 난타낸다.

Status	Value	Content
단말기 상태	0	정상
	1	잠김 상태
잠금 장치 제어 상태	0	닫힘 상태
	1	열림 상태
잠금장치 모니터링 상태	0	닫힘 상태
	1	열림 상태
	2	모니터링을 하지 않는 상태
단말기 뚜껑 열림 상태	0	닫힘 상태
	1	열림 상태

< 단말기 상태 정보 요약 >

2. Installation

2.1 시스템 요구 사항

■ **CPU**

Intel Pentium 133Mhz 이상

■ **Memory**

16M 이상

■ **USB Port**

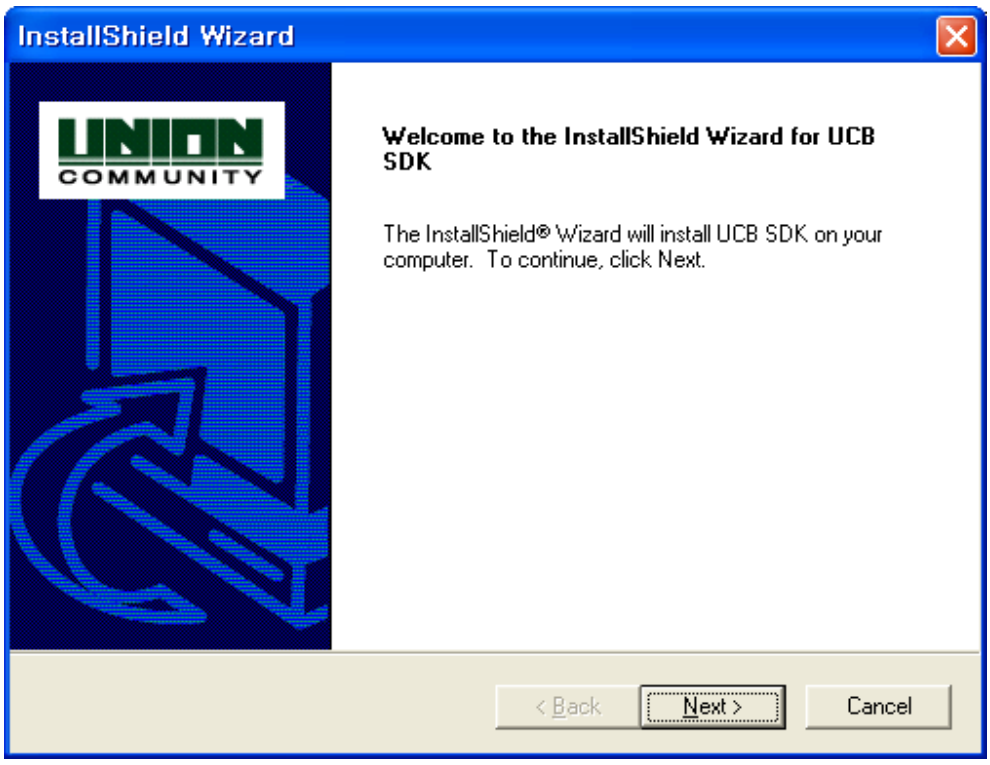
USB 1.1

■ **OS**

Windows 98/ME 또는 2000/XP/2003/Vista/Window 7

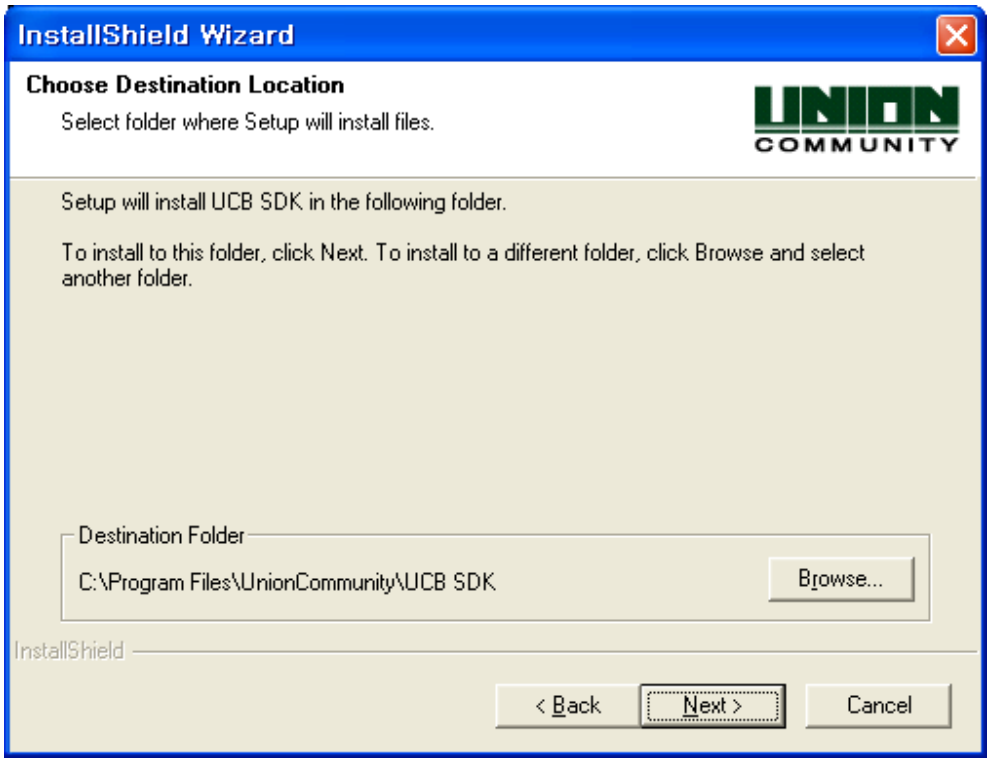
2.2 설치하기

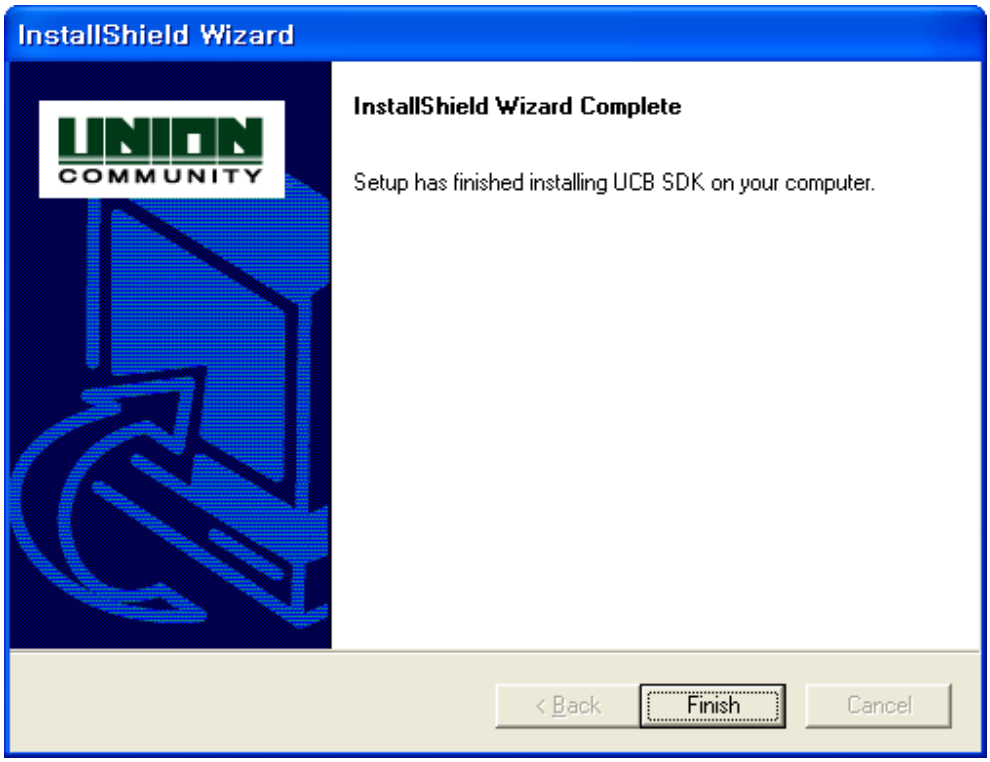
설치 CD를 삽입하면, Setup.exe가 자동으로 실행 됩니다.



단계별로 내용을 확인 하시면서 설치 하시기 바랍니다.

사용자 정보와 제품의 시리얼 번호를 입력 합니다.





2.3 설치되는 파일들

SDK 설치가 정상적으로 모두 끝나고 나면 지정한 설치 폴더에 다음과 같이 파일들이 설치 됩니다.

2.3.1 Windows System Directory

SDK의 사용을 위한 Core 모듈이 설치된다. 아래의 파일들이 설치된다.

UCSAPI40.dll

UCS SDK의 핵심 모듈. SDK의 모든 기능의 수행을 담당한다.

UCSAPICOM.dll

RAD Tool 개발자를 위한 COM 모듈.

WSEngine.dll

Windows Socket I/O 처리를 위한 통신 모듈

2.3.2 GAC(Global Assembly Cache) 폴더

.NET Framework 환경을 위한 Class Library가 설치되는 GAC폴더에 아래의 파일이 설치된다.
SDK의 설치 중에 .NET용 라이브러리를 설치할 경우 설치된다.

2.3.3 (설치폴더) \ Bin

SDK의 수행에 필요한 Core 파일 및 Sample용 실행 파일들이 들어있다.

UCSAPI40.dll / UCSAPICOM.dll / WSEngine.dll

Windows system32 폴더에 설치된 파일과 동일한 파일.

Demo application

UCS SDK의 기능을 간단히 테스트 해 볼 수 있는 다수의 Demo 프로그램이 들어있다.
Demo 프로그램 소스는 Samples 폴더에서 모두 제공된다.

2.3.4 (설치폴더) \ dotNET

SDK의 수행에 필요한 dotNET용 Class Library 파일들이 들어있다.

UNIONCOMM.SDK.UCSAPI40.dll

.NET용 Class Library 모듈. GAC에 설치되는 파일과 동일한 파일.

2.3.5 (설치폴더) \ dotNET \ Setup

.NET용 Class Library를 GAC에 설치하기 위한 설치파일이 들어있다.

Setup.exe (UCSAPI40.NET_Setup.msi)

.NET용 Class Library 설치 파일

2.3.6 (설치폴더) \ Inc

UCSAPI.h

UCS SDK의 메인 Header 파일로 이 파일을 include할 경우 내부적으로 UCSAPI_Basic.h, UCSAPI_Error.h, UCSAPI_Type.h 파일이 자동으로 포함된다.

UCSAPI_Basic.h

UCS SDK에서 사용되는 기본 데이터 타입을 정의하고 있다.

UCSAPI_Error.h

UCSAPI40 모듈에서 사용되는 에러 값을 정의하고 있다.

UCSAPI_Type.h

UCS SDK에서 사용되는 데이터 타입과 구조체 정보 등을 정의하고 있다.

2.3.7 (설치폴더) \ Lib

SDK를 이용해 VC++에서 개발하기 위한 Link용 Library 파일이 들어있다.

UCSAPI40.lib

VC++용으로 만들어진 Link용 Library 파일. VC++에서 UCBioBSP.dll을 정적으로 Link 할 때 사용된다.

2.3.8 (설치폴더) \ Samples

각 언어별 Sample source code가 폴더별로 구분되어 들어있다.

DLL

UCSAPI40.dll을 이용해 개발 가능한 Sample code가 들어있다.

1) VC6: Visual C++ 6.0용으로 만들어진 Sample이 들어있다.

COM

UCSAPICOM.dll을 이용해 개발 가능한 Sample code가 들어있다.

1) VB6: Visual Basic 6.0용으로 만들어진 Sample이 들어있다.

dotNET

UNIONCOMM.SDK.UCSAPI40dll을 이용해 Microsoft의 .NET 환경에서 개발 가능한 Sample code가 들어있다.

1) C#: VisualStudio.NET 2005, C#용으로 만들어진 Sample이 들어있다.

2.3.9 (설치폴더) \ Skins

각 언어별 Skin resource 파일이 포함되어 있다. 현재는 영문과 한글만 포함되어 있음.

3. API Reference for DLL

이 장에서는 DLL 모듈인 UCSAPI40.dll을 사용하기 위한 Type 및 API들에 대해 설명한다.

3.1 Type definitions

3.1.1 Basic types

UCSAPI_Basic에 선어되어 있으며 기본 Type들에 대해 정의하고 있다. OS나 CPU 독립적인 개발을 위해 기본 Type들에 대해 재정의 하고 있다. 아래 설명은 일반적인 Windows 상에서 C++로 개발하는 것을 기준으로 설명한다.

UCSAPI_SINT8 / UCSAPI_SINT16 / UCBioAPI_SINT32

부호 있는 1byte / 2bytes / 4bytes 값

UCSAPI_UINT8 / UCSAPI_UINT16 / UCBioAPI_UINT32

부호 없는 1byte / 2bytes / 4bytes 값

UCSAPI_SINT / UCSAPI_UINT

OS에 따라 달라지는 int / unsigned int 값. 32bits OS의 경우 4bytes로 동작하며 64bits OS의 경우에는 8bytes로 동작한다.

UCSAPI_VOID_PTR

Void*를 의미

UCSAPI_BOOL

UCSAPI_FALSE(0) / UCBioAPI_TRUE(1) 값을 가질 수 있다. Int와 동일하게 처리됨.

UCSAPI_CHAR / UCSAPI_CHAR_PTR

Char와 char*를 의미. 1byte 문자 및 문자열 값.

UCSAPI_NULL

NULL을 의미. ((void*)0)의 값으로 정의된다.

UCSAPI_HWND

윈도우의 Handle을 의미하는 HWND 값.

3.1.2 General types

UCAPI_Type.h에 선언되어 있으며 일반적인 Type들에 대해 정의 하고있다.

UCSAPI_RETURN

Prototype:

```
typedef UCSAPI_UINT32 UCSAPI_RETURN;
```

Description:

UCSAPI SDK의 함수들이 리턴하는 값을 정의. 일반적으로 UCS SDK의 오류값을 담게 된다. 자세한 오류 값들은 ERROR 정의를 참조.

UCSAPI_DATE_TIME_INFO

Prototype:

```
typedef struct ucsapi_datetime_info
{
    UCSAPI_UINT16    Year;
    UCSAPI_UINT8     Month;
    UCSAPI_UINT8     Day;
    UCSAPI_UINT8     Hour;
    UCSAPI_UINT8     Min;
    UCSAPI_UINT8     Sec;
    UCSAPI_UINT8     Reserved;
} UCSAPI_DATE_TIME_INFO, *UCSAPI_DATE_TIME_INFO_PTR;
```

Description:

날짜와 시간 정보를 담는구조체.

3.1.3 User information related types

UCAPI_Type.h에 선언되어 있으며 사용자 정보 관련 Type들에 대해 정의 하고있다.

UCSAPI_ACCESS_DATE_TYPE

Prototype:

```
typedef UCSAPI_UINT32          UCSAPI_ACCESS_DATE_TYPE;
```

Description:

UCSAPI_ACCESS_DATE 구조체의 데이터 타입을 정의한다. 출입 기간 데이터는 사용안함, 출입가능 기간, 출입 불가능 기간과 같이 총 3개의 데이터 타입을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

```
#define UCSAPI_DATE_TYPE_NOT_USE          0
#define UCSAPI_DATE_TYPE_ALLOW            1
#define UCSAPI_DATE_TYPE_RESTRICTION      2
```

UCSAPI_ACCESS_AUTHORITY

Prototype:

```
typedef struct ucsapi_access_authority
{
    UCSAPI_DATA_PTR             AccessGroup;
    UCSAPI_ACCESS_DATE_TYPE     AccessDateType;
    UCSAPI_ACCESS_DATE_PTR      AccessDate
} UCSAPI_ACCESS_AUTHORITY, UCSAPI_ACCESS_AUTHORITY_PTR;
```

Description:

사용자의 출입 권한 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

AccessGroup:

출입권한 그룹 Code 정보를 담고 있는 구조체에 대한 포인터.

AccessDateType:

UCSAPI_ACCESS_DATE 구조체가 가지는 데이터의 타입을 지정.

AccessDate:

출입 기간 정보를 담는 구조체에 대한 포인터.

UCSAPI_CARD_DATA

Prototype:

```
typedef struct ucsapi_card_data
{
        UCSAPI_UINT32           CardNum;
        UCSAPI_DATA_PTR         RFID[UCSAPI_CARD_NUMBER_MAX];
} UCSAPI_CARD_DATA, UCSAPI_CARD_DATA_PTR;
```

Description:

Card 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

CardNum:

총 RFID의 개수를 지정. 여기에 지정된 개수만큼 RFID 정보가 배열로 들어있다.

RFID:

RFID 정보를 담는 구조체의 포인터 배열.

UCSAPI_FINGER_DATA

Prototype:

```
typedef struct ucsapi_finger_data
{
        UCSAPI_UINT32           SecurityLevel;
        UCSAPI_BOOL             IsCheckSimilarFinger;
        UCSAPI_EXPORT_DATA_PTR  ExportData;
} UCSAPI_FINGER_DATA, UCSAPI_FINGER_DATA_PTR;
```

Description:

지문 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

SecurityLeve :

인증 시 사용할 보안 수준을 지정.

가질 수 있는 값은 UCBioBSP SDK의 UCBioAPI_FIR_SECURITY_LEVEL 정의 참조.

IsCheckSimilarFinger :

사용자 지문 데이터를 단말기로 추가 시 유사지문 체크 여부를 지정한다.

이 값을 True로 지정하는 경우 단말기는 등록 된 모든 사용자의 지문과 비교하여 유사지문이 존재 하는지 검사 하여 유사지문이 검출 되면 등록 실패 처리한다. 이 Flag는 단말기로 사용자 추가 작업을 느리게 함으로 지문 등록 사용자가 많은 경우 성능을 저하시키는 요인이 된다. UCSAPI_AddUserToTerminal 호출 시 사용된다.

ExportData:

변환된 Template 데이터가 담길 구조체의 포인터.

UCBioBSP SDK의 UCBioAPI_EXPORT_DATA 구조체 참조.

UCSAPI_AUTH_DATA

Prototype:

```
typedef struct ucsapi_auth_data
{
        UCSAPI_DATA_PTR         Password;
        UCSAPI_CARD_DATA_PTR    Card;
        UCSAPI_FINGER_DATA_PTR  Finger;
} UCSAPI_AUTH_DATA, UCSAPI_AUTH_DATA_PTR;
```

Description:

인증 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Password:

비밀번호 정보를 담는 구조체의 포인터.

Card :

Card 정보를 담는 구조체의 포인터.

Finger:

지문 정보를 담는 구조체의 포인터.

UCSAPI_PICTURE_HEADER

Prototype:

typedef struct ucsapi_picture_header

{

UCSAPI_UINT8 Format[4]; /* must be "jpg" */

UCSAPI_UINT32 Length; /* max length is 7 kbytes */

} UCSAPI_PICTURE_HEADER, UCSAPI_ PICTURE_HEADER_PTR;

Description:

사진 데이터의 Header 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Format :

사진 데이터의 포맷 정보를 가진다. (현재는 "JPG" 포맷만 지원)

파일의 확장자 값을 문자열로 지정한다.

Length:

사진 데이터의 크기 값을 가진다. 최대 지정할 수 있는 데이터는 7K 까지다.

UCSAPI_PICTURE_DATA

Prototype:

typedef struct ucsapi_picture_data

{

UCSAPI_PICTURE_HEADER Header;

UCSAPI_UINT8* Data;

} UCSAPI_PICTURE_DATA, UCSAPI_ PICTURE_DATA_PTR;

Description:

사진 데이터 정보를 담는 구조체.

Header :

사진 데이터의 Header 정보를 가진다.

Data :

UCSAPI_PICTURE_HEADER의 Format 형태의 이미지 데이터를 담는 버퍼의 포인터. (Binary stream). "JPG" 데이터의 Resolution은 320 * 240 이다.

UCSAPI_USER_COUNT

Prototype:

```
typedef struct ucsapi_user_count
{
        UCSAPI_UINT32 AdminNumber;
        UCSAPI_UINT32 UserNumber;
} UCSAPI_USER_COUNT, *UCSAPI_USER_COUNT_PTR;
```

Description:

단말기에 등록된 사용의 개수를 담는 구조체.
사용자는 관리자 와 일반 사용자로 구분되어 진다.
UCSAPI_GetUserCountFromTerminal 함수 호출 후 UCSAPI_CALLBACK_EVENT_GET_USER_COUNT 이벤트에서 얻을 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

AdminNumber:
등록된 관리자의 개수 값을 가진다.

UserNumber :
등록된 일반 사용자의 개수 값을 가진다.

UCSAPI_USER_INFO

Prototype:

```
typedef struct ucsapi_user_info
{
        UCSAPI_UINT32                   UserID;
        UCSAPI_DATA_PTR                 UserName;
        UCSAPI_DATA_PTR                 UniqueID;
        UCSAPI_USER_PROPERTY            Property;
        UCSAPI_UINT8                    Reserved[3];
        UCSAPI_ACCESS_AUTHORITY_PTR     AccessAuthority;
} UCSAPI_USER_INFO, UCSAPI_ USER_INFO_PTR;
```

Description:

사용자 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

UserID **:**

사용자 ID 정보를 가진다. 이 값은 최대 8자리의 숫자 데이터만 사용이 가능한다.

UserName **:**

사용자 이름 정보를 담는 구조체의 포인터. 이 값은 최대 UCSAPI_DATA_SIZE_USER_NAME 만큼 지정 할 수 있다.

UniqueID**:**

고유 번호(사원번호) 정보를 담는 구조체의 포인터. 이 값은 사용자 식별을 위해 UserID 대신 사용 할 수 있다. 최대 UCSAPI_DATA_SIZE_UNIQUE_ID 만큼 지정 할 수 있다.

Property**:**

사용자 속성(인증 타입 및 관리자 여부) 정보를 담는 구조체.

AccessAuthority:

출입권한 정보를 담는 구조체의 포인터.

UCSAPI_USER_DATA

Prototype:

```
typedef struct ucsapi_user_data
{
        UCSAPI_USER_INFO            UserInfo;
        UCSAPI_AUTH_DATA_PTR        AuthData;
        UCSAPI_PICTURE_DATA_PTR     PictureData;
} UCSAPI_USER_DATA, UCSAPI_ USER_DATA_PTR;
```

Description:

사용자 데이터를 담는 구조체. UCSAPI_AddUserToTerminal 호출 시 사용된다. 각각의 값들에 대한 설명은 다음과 같다.

UserInfo :

사용자 정보를 담는 구조체.

AuthData :

출입권한 데이터를 담은 구조체의 포인터.

PictureData :
사진 데이터를 담은 구조체의 포인터.

3.1.4 Log related types

UCAPI_Type.h에 선언되어 있으며 인증 로그 관련 Type들에 대해 정의 하고있다.

UCSAPI_GET_LOG_TYPE

Prototype:

```
typedef UCSAPI_UINT32 UCSAPI_GET_LOG_TYPE;
```

Description:

단말기로부터 로그 데이터를 얻어 오기 위하여 세가지 타입의 로그 종류를 정의 한다.
단말기의 모델에 따라 내부에 저장 할 수 있는 로그의 개수는 서로 차이가 있으며 최대 저장 용량을 초과시 이전의 기록을 일부 삭제하여 새로운 로그를 저장 하게 된다.
UCSAPI_GetAccessLogCountFromTerminal /
UCSAPI_GetAccessLogFromTerminal 호출 시 사용된다.

```
#define UCSAPI_GET_LOG_TYPE_NEW          0
#define UCSAPI_GET_LOG_TYPE_OLD          1
#define UCSAPI_GET_LOG_TYPE_ALL          2
```

UCSAPI_ACCESS_LOG_DATA

Prototype:

```
typedef struct ucsapi_access_log_data
{
    UCSAPI_UINT32               UserID;
    UCSAPI_DATE_TIME_INFO       DataTime;
    UCSAPI_UINT8                AuthMode;
    UCSAPI_UINT8                AuthType;
    UCSAPI_UINT8                Reserved[2];
    UCSAPI_BOOL                 IsAuthorized;
    UCSAPI_DATA_PTR             RFID;
    UCSAPI_PICTURE_DATA_PTR     PictureData;
```

} UCSAPI_ACCESS_LOG_DATA, UCSAPI_ACCESS_LOG_DATA_PTR;

Description:

인증 로그 데이터를 담는 구조체.
UCSAPI_GetAccessLogFromTerminal 호출 후 UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG 이벤트에서 얻을 수 있다.각각의 값들에 대한 설명은 다음과 같다.

UserID :

사용자 ID 값을 가진다.

***DateTime* :**

인증 시각 정보를 담은 구조체.

***AuthMode*:**

인증 모드 값을 가진다. UCSAPI_AUTH_MODE 정의 참조.

***AuthType*:**

인증 타입 값을 가진다. UCSAPI_AUTH_TYPE 정의 참조.

IsAuthorized:

인증 결과 값을 가진다. 성공일 경우 1의 값을 가지고 실패인 경우 0의 값을 가진다.

RFID:

Card 인증 시 사용된 RFID 데이터를 담은 구조체의 포인터.

PictureData:

인증 시 촬영된 사진 데이터를 담은 구조체의 포인터. 이 값은 사진 촬영이 가능한 단말기에서만 제공된다.

3.1.5 Callback related types

UCAPI_Type.h에 선언되어 있으며 Callback 이벤트의 Type들에 대해 정의 하고있다.
SDK는 단말기로부터 수신되는 데이터를 응용 프로그램으로 통지 하기 위해 Callback 함수를 사용한다. Callback 이벤트는 응용 프로그램의 요청에 대한 응답과 단말기로부터의 요청으로 구분 되어진다.

UCSAPI_CALLBACK_EVENT_CONNECTED

Prototype:

#define UCSAPI_CALLBACK_EVENT_CONNECTED UCSAPI_CALLBACK_EVENT+1

Description:

SDK 모듈은 단말기가 서버로 접속 되었을 때 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_DISCONNECTED

Prototype:

#define UCSAPI_CALLBACK_EVENT_DISCONNECTED UCSAPI_CALLBACK_EVENT+2

Description:

SDK 모듈은 단말기의 접속이 해지 되었을 때 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_REALTIME_ACCESS_LOG

Prototype:

#define UCSAPI_CALLBACK_EVENT_REALTIME_ACCESS_LOG UCSAPI_CALLBACK_EVENT+3

Description:

UCS SDK 모듈은 단말기가 S/N 모드로 동작하고 단말기에서 사용자 인증을 수행 하였을 때 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG UCSAPI_CALLBACK_EVENT+4

Description:

SDK 모듈은 응용프로그램의 단말기에 저장되어있는 인증로그 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG_COUNT

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG_COUNT UCSAPI_CALLBACK_EVENT+5

Description:

SDK 모듈은 응용프로그램의 단말기에 저장되어있는 인증로그 개수 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_ADD_USER

Prototype:

#define UCSAPI_CALLBACK_EVENT_ADD_USER UCSAPI_CALLBACK_EVENT+10

Description:

SDK 모듈은 응용프로그램의 사용자 추가 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_DELETE_USER

Prototype:

#define UCSAPI_CALLBACK_EVENT_DELETE_USER UCSAPI_CALLBACK_EVENT+11

Description:

SDK 모듈은 응용프로그램의 사용자 삭제 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_DELETE_ALL_USER

Prototype:

#define UCSAPI_CALLBACK_EVENT_DELETE_ALL_USER UCSAPI_CALLBACK_EVENT+12

Description:

SDK 모듈은 응용프로그램의 모든 사용자 삭제 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_GET_USER_COUNT

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_USER_COUNT UCSAPI_CALLBACK_EVENT+13

Description:

SDK 모듈은 응용프로그램의 단말기에 저장되어있는 사용자 개수 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_GET_USER_INFO_LIST

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_USER_INFO_LIST UCSAPI_CALLBACK_EVENT+14

Description:

SDK 모듈은 응용프로그램의 단말기에 저장되어있는 사용자 리스트 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_GET_USER_DATA

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_USER_DATA UCSAPI_CALLBACK_EVENT+15

Description:

SDK 모듈은 응용프로그램의 단말기에 저장되어있는 사용자 데이터 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_VERIFY_USER_AUTH_INFO

Prototype:

#define UCSAPI_CALLBACK_EVENT_VERIFY_USER_AUTH_INFO UCSAPI_CALLBACK_EVENT+20

Description:

단말기는 1:1 서버 인증 시 인증에 필요한 정보(Authentication Type)를 획득하기 위해 응용 프로그램으로 이 이벤트를 통지한다. 이때 응용 프로그램은 사용자의 인증 타입 정보를 단말기로 전송 해야 한다. 이 이벤트는 서버에서 인증을 수행 할 때만 발생하며, 항상 다음의 이벤트 전에 발행하게 된다.

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1,

UCSAPI_CALLBACK_EVENT_VERIFY_CARD,

UCSAPI_CALLBACK_EVENT_VERIFY_PASSWORD

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1

Prototype:

#define UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1 UCSAPI_CALLBACK_EVENT+21

Description:

단말기는 1:1 지문 인증 시 이 이벤트를 응용 프로그램으로 통지 한다. 이때 응용 프로그램은 인증 결과를 단말기로 전송 해야 한다. 이 이벤트는 서버에서 인증을 수행 할 때만 발생한다.

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_N

Prototype:

#define UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_N UCSAPI_CALLBACK_EVENT+22

Description:

단말기는 1:N 지문 인증 시 이 이벤트를 응용프로그램으로 통지 한다. 이때 응용 프로그램은 인증 결과를 단말기로 전송 해야 한다. 이 이벤트는 서버에서 인증을 수행 할 때만 발생한다.

UCSAPI_CALLBACK_EVENT_VERIFY_CARD

Prototype:

#define UCSAPI_CALLBACK_EVENT_VERIFY_CARD UCSAPI_CALLBACK_EVENT+23

Description:

단말기는 Card 인증 시 이 이벤트를 응용 프로그램으로 통지 한다. 이때 응용 프로그램은 인증 결과를 단말기로 전송 하여야 한다. 이 이벤트는 서버에서 인증을 수행 할 때만 발생한다.

UCSAPI_CALLBACK_EVENT_VERIFY_PASSWORD

Prototype:

#define UCSAPI_CALLBACK_EVENT_VERIFY_PASSWORD UCSAPI_CALLBACK_EVENT+24

Description:

단말기는 Password 인증 시 이 이벤트를 응용 프로그램으로 통지 한다. 이때 응용 프로그램은 인증 결과를 단말기로 전송 하여야 한다. 이 이벤트는 서버에서 인증을 수행 할 때만 발생한다.

UCSAPI_CALLBACK_EVENT_GET_TERMINAL_OPTION

Prototype:

#define UCSAPI_CALLBACK_EVENT_GET_TERMINAL_OPTON UCSAPI_CALLBACK_EVENT+30

Description:

SDK 모듈은 응용프로그램의 단말기 옵션 설정정보 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_SET_TERMINAL_OPTION

Prototype:

#define UCSAPI_CALLBACK_EVENT_SET_TERMINAL_OPTION UCSAPI_CALLBACK_EVENT+31

Description:

SDK 모듈은 응용프로그램의 단말기 옵션 설정 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_FW_UPGRADING

Prototype:

#define UCSAPI_CALLBACK_EVENT_FW_UPGRADING UCSAPI_CALLBACK_EVENT+80

Description:

SDK 모듈은 응용프로그램의 펌웨어 업그레이드 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다. Upgrade의 진행 상황을 응용 프로그램으로 통지 하기 위한 이벤트 이다.

UCSAPI_CALLBACK_EVENT_FW_UPGRADE

Prototype:

#define UCSAPI_CALLBACK_EVENT_FW_UPGRADED UCSAPI_CALLBACK_EVENT+81

Description:

SDK 모듈은 응용 프로그램의 펌웨어 업그레이드 요청에 대한 응답으로 업그레이드 완료를 응용프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_FW_VERSION

Prototype:

#define UCSAPI_CALLBACK_EVENT_FW_VERSION UCSAPI_CALLBACK_EVENT+82

Description:

SDK 모듈은 응용 프로그램의 펌웨어 버전 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_OPEN_DOOR

Prototype:

#define UCSAPI_CALLBACK_EVENT_OPEN_DOOR UCSAPI_CALLBACK_EVENT+90

Description:

SDK 모듈은 응용 프로그램의 출입문 일시 개방 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_TERMINAL_STATUS

Prototype:

#define UCSAPI_CALLBACK_EVENT_TERMINAL_STATUS UCSAPI_CALLBACK_EVENT+91

Description:

SDK 모듈은 단말기의 상태 및 단말기 주변에 부착된 장치의 상태 정보를 주기적 또는 상태 변화가 있을 경우 즉시 서버로 그 상태를 통지 한다.

UCSAPI_CALLBACK_EVENT_PICTURE_LOG

Prototype:

#define UCSAPI_CALLBACK_EVENT_PICTURE_LOG UCSAPI_CALLBACK_EVENT+100

Description:

SDK 모듈은 단말기로부터 Picture 로그를 수신 한 경우 Picture 로그 데이터를 응용 프로그램으로 통지 한다. 단말기 인증 방식으로 인증 한경우 Picture 로그는 UCSAPI_CALLBACK_EVENT_REALTIME_ACCESS_LOG 이벤트와 함께 응용 프로그램로 통지 되지만 서버 인증 방식으로 인증 하는 경우 단말기는 응용 프로그램으로부터 인증 결과를 수신 후에 Picture 로그를 별도로 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_ANTIPASSBACK

Prototype:

#define UCSAPI_CALLBACK_EVENT_ANTIPASSBACK UCSAPI_CALLBACK_EVENT+101

Description:

단말기는 Antipassback 사용 옵션이 설정되어 있는 경우 사용자 인증시 사용자의 현재 Antipassback 상태를 얻기 위해 응용 프로그램으로 이 이벤트를 통지 한다. 이 때 응용 프로그램은 사용자의 Antipassback 상태를 즉시 단말기로 전송해야 한다. 이 이벤트는 단말기가 단말기에서 인증을 수행 할 때만 발생한다.

UCSAPI_CALLBACK_EVENT_SET_ACCESS_CONTROL_DATA

Prototype:

#define UCSAPI_CALLBACK_EVENT_SET_ACCESS_CONTROL_DATA
UCSAPI_CALLBACK_EVENT+102

Description:

SDK 모듈은 응용 프로그램의 단말기 출입권한 설정 요청에 대한 응답으로 이 이벤트를 응용 프로그램으로 통지 한다.

UCSAPI_CALLBACK_EVENT_TYPE

Prototype:

typedef UCSAPI_UINT32 UCSAPI_CALLBACK_EVENT_TYPE

Description:

UCSAPI_CALLBACK_EVENT_HANDLER의 두번째 인자 값을 정의 한다.

UCSAPI_CALLBACK_EVENT_HANDLER

Prototype:

typedef UCSAPI_RETURN (UCSAPI * UCSAPI_CALLBACK_EVENT_HANDLER) (
UCSAPI_UINT32 TerminalID, UCSAPI_UINT32 EventType,
UCSAPI_UINT32 wParam, UCSAPI_UINT32 lParam);

Description:

단말기로부터 발생하는 이벤트를 받기 위한 콜 백 함수에 대한 정의이다.

UCSAPI_CALLBACK_PARAM_0

Prototype:

typedef struct ucsapi_callback_param_0

```
{
        UCSAPI_UINT32               ClientID;
        UCSAPI_UINT32               ErrorCode;
        UCSAPI_PROGRESS_INFO        Progress;
} UCSAPI_CALLBACK_PARAM_0, *UCSAPI_CALLBACK_PARAM_0_PTR;
```

Description:

UCSAPI_CALLBACK_EVENT_HANDLER의 세번째 인자로 넘어도는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

ClientID :

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

ErrorCode :

실행한 작업 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

Progress :

실행한 작업의 진행 정보를 담은 구조체.
UCSAPI_GetUserInfoListFromTerminal/UCSAPI_GetAccessLogFromTerminal/
USCAPI_UpgradeFirmwareToTerminal 함수 호출 후 얻을 수 있다.

UCSAPI_CALLBACK_DATA_TYPE

Prototype:

```
typedef UCSAPI_UINT32 UCSAPI_CALLBACK_DATA_TYPE;
```

Description:

UCSAPI_CALLBACK_PARAM_1 구조체의 데이터 타입을 지정한다.

```
#define UCSAPI_CALLBACK_DATA_TYPE_USER_INFO         0
#define UCSAPI_CALLBACK_DATA_TYPE_USER_DATA         1
#define UCSAPI_CALLBACK_DATA_TYPE_ACCESS_LOG        2
```

UCSAPI_CALLBACK_PARAM_1

Prototype:

```
typedef struct ucsapi_callback_param_1
{
        UCSAPI_CALLBACK_DATA_TYPE      DataType;
        Union {
                UCSAPI_USER_INFO_PTR            UserInfo;
                UCSAPI_USER_DATA_PTR            UserData;
                UCSAPI_ACCESS_LOG_DATA_PTR      AccessLog;
              } Data;
} UCSAPI_CALLBACK_PARAM_1, *UCSAPI_CALLBACK_PARAM_1_PTR;
```

Description:

UCSAPI_CALLBACK_EVENT_HANDLER의 네번째 인자로 넘어도는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

DataType:

이 구조체가 가지는 데이터의 타입을 지정. UCSAPI_CALLBACK_DATA_TYPE 참조.

Data:

실제 데이터를 지정하는 union 구조체. UserInfo, UserData, AccessLog의 값을 하나의 동일 주소 포인터로 저장해 사용 할 수 있게 되어 있다.

UCSAPI_PROGRESS_INFO

Prototype:

```
typedef struct ucsapi_progress_info
{
        UCSAPI_UINT32           CurrentIndex;
        UCSAPI_UINT32           TotalNumber;
} UCSAPI_PROGRESS_INFO, *UCSAPI_PROGRESS_INFO_PTR;
```

Description:

실행한 작업의 진행 정보를 담은 구조체. UCS SDK는 여러 개의 레코드를 응용 프로그램으로 통지 할 때 이 구조체에 진행 정보를 담아 UCSAPI_CALLBACK_PARAM_0 구조체와 함께 응용 프로그램으로 통지한다.

UCSAPI_GetUserInfoListFromTerminal/UCSAPI_GetAccessLogFromTerminal/
USCAPI_UpgradeFirmwareToTerminal 함수 호출 후 얻을 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

CurrentIndex:
현재 전송 중인 레코드의 인덱스

TotalNumber :
전송 해야 할 총 레코드의 개수

3.1.6 Access control setting related types

UCAPI_Type.h에 선언되어 있으며 단말기 출입제어 관련 Type들에 대해 정의 하고있다.

UCSAPI_TIMEZONE

Prototype:

```
typedef struct ucsapi_timezone
{
        UCSAPI_TIME_HH_MM       StartTime;
        UCSAPI_TIME_HH_MM       EndTime;
} UCSAPI_TIMEZONE, * UCSAPI_TIMEZONE_PTR;
```

Description:

시간대 정보를 담은 구조체.

StratTime / EndTime:

시작부터 종료까지의 시간 정보를 담은 구조체.

UCSAPI_ACCESS_TIMEZONE

Prototype:

```
typedef struct ucsapi_access_timezone
{
        UCSAPI_CHAR             Code[UCSAPI_DATA_SIZE_CODE4];
        UCSAPI_TIMEZONE         Zone[12];
        UCSAPI_UINT8            Reserved[4]
} UCSAPI_ACCESS_TIMEZONE, * UCSAPI_ACCESS_TIMEZONE_PTR;
```

Description:

하루 중 출입가능한 시간대 정보를 담은 구조체.

하나의 시간 코드에 지정 가능한 시간대는 최대 12개 이다. 각각의 값들에 대한 설명은 다음과 같다.

Code:

시간대의 식별자 코드 값으로 고정 UCSAPI_DATA_SIZE_CODE 크기의 문자열.

Zone

시간대 정보를 담은 구조체 배열.

UCSAPI_ACCESS_TIMEZONE_DATA

Prototype:

```
typedef struct ucsapi_access_timezone_data
{
        UCSAPI_UINT32                   TimezoneNum;
        UCSAPI_ACCESS_TIMEZONE          Timezone[UCSAPI_ACCESS_TIMEZONE_MAX];
} UCSAPI_ACCESS_TIMEZONE_DATA, * UCSAPI_ACCESS_TIMEZONE_DATA_PTR;
```

Description:

출입가능시간대 데이터를 담은 구조체. 최대 128개의 시간대 코드 정보를 지정 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

TimezoneNum:

총 출입가능시간대 코드의 개수를 지정. 여기에 지정된 개수만큼 시간대 정보가 배열로 들어있다.

Timezone:

출입가능시간대 코드 정보를 담는 구조체 배열.

UCSAPI_ACCESS_HOLIDAY

Prototype:

```
typedef struct ucsapi_access_holiday
{
```

```
    UCSAPI_CHAR                 Code[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_DATE_MM_DD           Date[32];
} UCSAPI_ACCESS_HOLIDAY, * UCSAPI_ACCESS_HOLIDAY_PTR;
```

Description:

휴일 정보를 담은 구조체.

휴일 코드에 지정 가능한 휴일은 최대 32개 까지 지정 할 수 이다. 각각의 값들에 대한 설명은 다음과 같다.

Code:

휴일의 식별자 코드 값으로 고정 UCSAPI_DATA_SIZE_CODE 크기의 문자열.

Date

휴일 정보를 담은 구조체 배열이다.

UCSAPI_ACCESS_HOLIDAY_DATA

Prototype:

```
typedef struct ucsapi_access_holiday_data
{
    UCSAPI_UINT32               HolidayNum;
    UCSAPI_ACCESS_TIMEZONE      Holiday[UCSAPI_ACCESS_HOLIDAY_MAX];
} UCSAPI_ACCESS_HOLIDAY_DATA, * UCSAPI_ACCESS_HOLIDAY_DATA_PTR;
```

Description:

휴일 데이터를 담은 구조체.

최대 64개의 휴일 코드 데이터를 지정 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

HolidayNum:

총 휴일 코드의 개수를 지정. 여기에 지정된 개수만큼 Holiday 정보가 배열로 들어있다.

Holiday:

휴일 코드 정보를 담은 구조체 배열이다.

UCSAPI_ACCESS_TIMEZONE_CODE

Prototype:

```
typedef struct ucsapi_access_timezone_code
{
    UCSAPI_CHAR     Sun[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Mon[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Tue[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Wed[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Thu[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Fri[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Sat[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_CHAR     Hol[UCSAPI_DATA_SIZE_CODE4];
} UCSAPI_ACCESS_TIMEZONE_CODE, * UCSAPI_ACCESS_TIMEZONE_CODE_PTR;
```

Description:

요일 별 출입가능시간대 코드 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Sun / Mon / Tue / Wed / Thu / Fri / Sat:

인증 시 사용 할 요일별 출입가능시간대 코드 값을 가진다.

Hol:

인증 시 휴일에 적용할 출입가능시간대 코드 값을 가진다.

UCSAPI_ACCESS_TIME

Prototype:

```
typedef struct ucsapi_access_time
{
    UCSAPI_CHAR                 Code[UCSAPI_DATA_SIZE_CODE4];
    UCSAPI_ACCESS_TIMEZONE_CODE Timezone;
    UCSAPI_CHAR                 Holiday[UCSAPI_DATA_SIZE_CODE4];
} UCSAPI_ACCESS_TIME, * UCSAPI_ACCESS_TIME_PTR;
```

Description:

출입가능시간 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Code:

출입가능시간의 식별자 코드 값으로 고정 UCSAPI_DATA_SIZE_CODE 크기의 문자열.

Timezone:

요일 별 출입가능시간대 코드 값을 가진다.

Holiday:

출입가능시간 코드에서 사용 할 휴일 코드 값을 가진다.
여기서 지정된 휴일 코드는 UCSAPI_ACCESS_TIMEZONE_CODE 구조체의 Hol에서 지정한 시간대의 적용을 받는다.

UCSAPI_ACCESS_TIME_DATA

Prototype:

```
typedef struct ucsapi_access_time_data
{
        UCSAPI_UINT32                   AccessTimeNum;
        UCSAPI_ACCESS_TIME              AccessTime[UCSAPI_ACCESS_TIME_MAX];
} UCSAPI_ACCESS_TIME_DATA, * UCSAPI_ACCESS_TIME_DATA_PTR;
```

Description:

출입가능시간 데이터를 담은 구조체.
최대 128개의 출입가능시간 코드 정보를 지정 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

AccessTimeNum:

총 출입가능시간 코드의 개수를 지정. 여기에 지정된 개수만큼 AccessTime 정보가 배열로 들어있다.

AccessTime:

인증 가능시간 Code 정보를 담는 구조체 배열이다.

UCSAPI_ACCESS_GROUP

Prototype:

```
typedef struct ucsapi_access_group
```

```
{
        UCSAPI_CHAR             Code[UCSAPI_DATA_SIZE_CODE4];
        UCSAPI_CHAR             AccessTime1[UCSAPI_DATA_SIZE_CODE4];
        UCSAPI_CHAR             AccessTime2[UCSAPI_DATA_SIZE_CODE4];
        UCSAPI_CHAR             AccessTime3[UCSAPI_DATA_SIZE_CODE4];
        UCSAPI_CHAR             AccessTime4[UCSAPI_DATA_SIZE_CODE4];
} UCSAPI_ACCESS_GROUP, * UCSAPI_ACCESS_GROUP_PTR;
```

Description:

출입그룹 코드 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.
출입그룹 코드에는 최대 4개의 출입가능시간 코드를 지정 할 수 있다.

Code:

출입그룹의 식별자 코드 값으로 고정 UCSAPI_DATA_SIZE_CODE 크기의 문자열.

AccessTime1 / AccessTime2 / AccessTime3 / AccessTime4:

출입그룹에서 사용 할 출입가능시간 코드 정보를 가진다.

UCSAPI_ACCESS_GROUP_DATA

Prototype:

```
typedef struct ucsapi_access_group_data
{
        UCSAPI_UINT32           AccessGroupNum;
        UCSAPI_ACCESS_GROUP     AccessGroup[UCSAPI_ACCESS_GROUP_MAX];
} UCSAPI_ACCESS_GROUP_DATA, * UCSAPI_ACCESS_GROUP_DATA_PTR;
```

Description:

출입그룹 데이터를 담은 구조체. 최대 128개의 출입그룹 코드 정보를 지정 할 수 있다.
각각의 값들에 대한 설명은 다음과 같다.

AccessGroupNum:

총 출입그룹 코드의 개수를 지정. 여기에 지정된 개수만큼 AccessGroup 정보가 배열로 들어있다.

AccessGroup:

출입그룹 코드 정보를 담은 구조체 배열이다.

UCSAPI_ACCESS_CONTROL_DATA_TYPE

Prototype:

typedef UCSAPI_UINT32 UCSAPI_ACCESS_CONTROL_DATA_TYPE

#define UCSAPI_ACCESS_CONTROL_DATA_TYPE_TIMEZONE 0
#define UCSAPI_ACCESS_CONTROL_DATA_TYPE_HOLIDAY 1
#define UCSAPI_ACCESS_CONTROL_DATA_TYPE_TIME 2
#define UCSAPI_ACCESS_CONTROL_DATA_TYPE_GROUP 3

Description:

UCSAPI_ACCESS_CONTROL_DATA 구조체가 가지는 데이터의 타입을 지정.

UCSAPI_ACCESS_CONTROL_DATA

Prototype:

```
typedef struct ucsapi_access_control_data
{
    UCSAPI_ACCESS_CONTROL_DATA_TYPE    DataType;
    union {
        UCSAPI_ACCESS_TIMEZONE_DATA_PTR    Timezone;
        UCSAPI_ACCESS_HOLIDAY_DATA_PTR     Holiday;
        UCSAPI_ACCESS_TIME_DATA_PTR        AccessTime;
        UCSAPI_ACCESS_GROUP_DATA_PTR       AccessGroup;
    } Data;
} UCSAPI_ACCESS_CONTROL_DATA, * UCSAPI_ACCESS_CONTROL_DATA_PTR;
```

Description:

출입제어 정보를 담은 구조체. 구조체. Timezone, Holiday, AccessTime, AccessGroup의 값을 하나의 동일 주소 포인터로 저장해 사용 할 수 있게 되어 있다
UCSAPI_SetAccessControlDataToTerminal 함수에서 사용된다.
각각의 값들에 대한 설명은 다음과 같다.

DataType:

이 구조체가 가지는 데이터의 타입을 지정. UCSAPI_ACCESS_CONTROL_DATA_TYPE 참조..

3.1.7 Authentication related types

UCAPI_Type.h에 선언되어 있으며 사용자 인증 관련 타입들에 대해 정의 하고있다.

UCSAPI_AUTH_TYPE

Prototype:

typedef UCSAPI_UINT32 UCSAPI_AUTH_TYPE;

Description:

사용자 인증 시 인증 타입을 정의한다.

#define UCSAPI_AUTH_TYPE_FINGER_1_TO_N	0
#define UCSAPI_AUTH_TYPE_FINGER_1_TO_1	1
#define UCSAPI_AUTH_TYPE_FINGER_CARD	2
#define UCSAPI_AUTH_TYPE_CARD	3
#define UCSAPI_AUTH_TYPE_PASSWORD	4

UCSAPI_AUTH_MODE

Prototype:

typedef UCSAPI_UINT32 UCSAPI_AUTH_MODE;

Description:

사용자 인증 시 인증 모드를 정의한다. 인증 모드는 인증시 그 목적을 정의 한다. 일반 적으로 단말기를 근태용으로 사용 시 적용 할 수 있다.

#define UCSAPI_AUTH_MODE_ATTENDANCE	0
#define UCSAPI_AUTH_MODE_LEAVE	1
#define UCSAPI_AUTH_MODE_NORMAL	2
#define UCSAPI_AUTH_MODE_OUT	3
#define UCSAPI_AUTH_MODE_RETURN	4

UCSAPI_INPUT_DATA_CARD

Prototype:

```
typedef struct ucsapi_input_data_card
{
        UCSAPI_UINT32           AuthMode;
        UCSAPI_DATA             RFID;
} UCSAPI_INPUT_DATA_CARD, *UCSAPI_INPUT_DATA_CARD_PTR;
```

Description:

단말기에서 Card 인증 시 입력된 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

AuthMode:

단말기에서 입력된 인증 모드 값을 가진다. UCSAPI_AUTH_MODE 참조.

RFID:

단말기에서 입력된 RFID 정보를 담은 구조체.

UCSAPI_INPUT_DATA_PASSWORD

Prototype:

```
typedef struct ucsapi_input_data_password
{
        UCSAPI_UINT32           UserID;
        UCSAPI_UINT32           AuthMode;
        UCSAPI_DATA             Password;
} UCSAPI_INPUT_DATA_PASSWORD, *UCSAPI_INPUT_DATA_PASSWORD_PTR;
```

Description:

단말기에서 비밀번호 인증 시 입력된 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

UserID:

사용자 ID 정보를 가진다.

AuthMode:

인증 모드 값을 가진다. UCSAPI_AUTH_MODE 참조.

Password:

비밀번호 정보를 담은 구조체.

UCSAPI_INPUT_DATA_FINGER_1_TO_1

Prototype:

```
typedef struct ucsapi_input_data_finger_1_to_n
{
        UCSAPI_UINT32           UserID;
        UCSAPI_UINT32           AuthMode;
        UCSAPI_UINT32           SecurityLevel;
        UCSAPI_DATA             Finger;
} UCSAPI_INPUT_DATA_FINGER_1_TO_1, *UCSAPI_INPUT_DATA_FINGER_1_TO_1_PTR;
```

Description:

단말기에서 1:1 지문 인증 시 입력된 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

UserID:

사용자 ID 정보를 가진다.

AuthMode:

인증 모드 값을 가진다. UCSAPI_AUTH_MODE 참조.

SecurityLevel:

인증 시 사용할 보안 레벨 값을 가진다.

가질 수 있는 값은 UCBioBSP SDK의 UCBioAPI_FIR_SECURITY_LEVEL 정의 참조.

Finger:

지문 정보를 담은 구조체.

UCSAPI_INPUT_DATA_FINGER_1_TO_N

Prototype:

```
typedef struct ucsapi_input_data_finger_1_to_n
{
        UCSAPI_UINT32           AuthMode;
        UCSAPI_UINT32           SecurityLevel;
        UCSAPI_UINT32           InputIDLength;
        UCSAPI_DATA             Finger;
} UCSAPI_INPUT_DATA_FINGER_1_TO_N, *UCSAPI_INPUT_DATA_FINGER_1_TO_N_PTR;
```

Description:

단말기에서 1:N 지문 인증 시 입력된 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

UserID:

사용자 ID 정보를 가진다.

AuthMode:

인증 모드 값을 가진다. UCSAPI_AUTH_MODE 참조.

SecurityLevel:

인증 시 사용되어질 보안 레벨 값을 가진다.

InputIDLength:

단말기에서 입력된 ID의 길이 값을 가진다. 이 값은 1:N 지문 인증 시 인증 범위를 축소하여 인증 속도를 개선 하고자 할 때 사용될 수 있다. 등록 사용자의 ID 범위가 0001~1000의 범위를 가질경우 UserID의 값이 5이고 InputIDLength의 값이 2이면 인증 ID 범위는 0500~1000이 된다.

Finger:

단말기에서 입력된 지문 정보를 담은 구조체.

UCSAPI_INPUT_DATA_TYPE

Prototype:

```
typedef UCSAPI_UINT32           UCSAPI_INPUT_DATA_TYPE;
```

```
#define UCSAPI_INPUT_DATA_TYPE_FINGER_1_TO_N        0
#define UCSAPI_INPUT_DATA_TYPE_FINGER_1_TO_1        1
#define UCSAPI_INPUT_DATA_TYPE_PASSWORD             2
#define UCSAPI_INPUT_DATA_TYPE_CARD                 3
#define UCSAPI_INPUT_DATA_TYPE_FINGER_CARD          4
```

Description:

UCSAPI_INPUT_DATA_TYPE 구조체가 가지는 데이터의 type을 지정.

UCSAPI_INPUT_DATA

Prototype:

```
typedef struct ucsapi_input_data
{
        UCSAPI_ANTIPASSBACK_LEVEL       AntipassbackLevel;
        UCSAPI_INPUT_DATA_TYPE          DataType;
        Union {
                UCSAPI_INPUT_DATA_FINGER_1_TO_1_PTR   Finger1To1;
                UCSAPI_INPUT_DATA_FINGER_1_TO_N_PTR   Finger1ToN;
                UCSAPI_INPUT_DATA_CARD_PTR            Card;
                UCSAPI_INPUT_DATA_PASSWORD_PTR        Password;
        } Data;
} UCSAPI_INPUT_DATA, *UCSAPI_INPUT_DATA_PTR;
```

Description:

사용자 인증 시 단말기로부터 입력된 정보를 담은 구조체. 단말기가 서버 인증 모드를 사용 시 이 구조체에 입력 정보를 저장하여 응용 프로그램으로 인증을 요청 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

AntipassbackLevel:

단말기에 설정되어 있는 안티패스백 레벨 값을 가진다. 응용 프로그램은 안티패스백 기능 구현 시 이 값을 참조 할 수 있다.

DataType:

이 구조체가 가지는 데이터의 타입을 지정. UCSAPI_INPUT_DATA_TYPE 참조.

Data:

실제 데이터를 지정하는 union 구조체. Finger1To1, Finger1ToN, Card, Password의 값을 하나의 동일 주소 포인터로 저장해 사용 할 수 있게 되어 있다.

UCSAPI_INPUT_ID_TYPE

Prototype:

```
typedef UCSAPI_UINT32            UCSAPI_INPUT_ID_TYPE;

#define UCSAPI_INPUT_ID_TYPE_USER_ID                0
#define UCSAPI_INPUT_ID_TYPE_UNIQUE_ID              1
#define UCSAPI_INPUT_ID_TYPE_RFID                   2
```

Description:

단말기에서 입력된 ID의 타입을 지정. 1:1 인증 시 입력되는 ID 타입은 단말기 옵션 설정에 변경 할 수 있다. 기본 값으로는 UCSAPI_INPUT_ID_TYPE_USER_ID 타입을 사용한다. 사용자 ID 값의 최대 사용할 수 있는 범위는 숫자 8자리까지 사용할 수 있기 때문에 이를 초과시 UCSAPI_INPUT_ID_TYPE_UNIQUE_ID 타입을 사용하면 최대 20자리 까지 사용이 가능해 진다.

UCSAPI_INPUT_ID_DATA

Prototype:

```
typedef struct ucsapi_input_id_data
{
        UCSAPI_INPUT_ID_TYPE        DataType;
        Union {
                UCSAPI_UINT32*      UserID;
                UCSAPI_DATA_PTR     UniqueID
                UCSAPI_DATA_PTR     RFID;
        } Data;
} UCSAPI_INPUT_ID_DATA, *UCSAPI_INPUT_ID_DATA_PTR;
```

Description:

서버에서 사용자 인증 시 단말기로부터 입력된 ID 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

DataType:

이 구조체가 가지는 데이터의 타입을 지정. UCSAPI_INPUT_ID_TYPE 참조.

Data:

실제 데이터를 지정하는 union 구조체. UserID, UniqueID, RFID 값을 하나의 동일 주소 포인터로 저장해 사용 할 수 있게 되어 있다.

UCSAPI_AUTH_INFO

Prototype:

```
typedef struct ucsapi_auth_info
{
        UCSAPI_UINT32               UserID;
        UCSAPI_BOOL                 IsAccessibility;
        UCSAPI_USER_PROPERTY        Property;
        UCSAPI_UINT32               ErrorCode;
} UCSAPI_AUTH_INFO, *UCSAPI_AUTH_INFO_PTR;
```

Description:

사용자의 인증 정보를 담는 구조체.UCSAPI_SendAuthInfoToTerminal 함수에서 사용된다.
각각의 값들에 대한 설명은 다음과 같다.

UserID:

사용자의 ID 값을 가진다.

IsAccessibility:

사용자의 인증 가능 여부 값을 가진다.

Property:

사용자 속성(인증 타입, 관리자) 정보를 담은 구조체.

ErrorCode:

사용자가 인증 권한이 없는 경우 그에 대한 에러코드 값을 가진다. 에러코드 표 참조.

UCSAPI_AUTH_RESULT

Prototype:

```
typedef struct ucsapi_auth_result
{
        UCSAPI_UINT32               UserID;
        UCSAPI_BOOL                 IsAuthorized;
        UCSAPI_BOOL                 IsVistor;
        UCSAPI_DATE_TIME_INFO       AuthorizedTime;
        UCSAPI_UINT32               ErrorCode;
} UCSAPI_INPUT_DATA, *UCSAPI_INPUT_DATA_PTR;
```

Description:

사용자의 인증 결과 정보를 담은 구조체.
UCSAPI_SendAuthResultToTerminal 함수에서 사용된다.

UserID:

인증 되거나 또는 인증 시도한 사용자의 ID 값을 가진다.

IsAuthorized:

인증 결과 값을 가진다.

IsVisitor:

인증된 사용자의 방문객 여부 값은 가진다.

AuthorizedTime:

인증 시각 정보를 담은 구조체.

ErrorCode:

인증 실패 시 그에 대한 에러코드 값을 가진다. 에러코드 표 참조.

3.1.8 Terminal option setting related types

UCAPI_Type.h에 선언되어 있으며 단말기 옵셜 설정 관련 Type들에 대해 정의 하고있다.

UCSAPI_TERMINAL_TIMEZONE

Prototype:

```
typedef struct ucsapi_terminal_timezone
{
        UCSAPI_UINT8        IsUsed;
        UCSAPI_UINT8        StartHour;
        UCSAPI_UINT8        StartMin;
        UCSAPI_UINT8        EndHour;
        UCSAPI_UINT8        EndMin;
} UCSAPI_TERMINAL_TIMEZONE, * UCSAPI_TERMINAL_TIMEZONE_PTR;
```

Description:

단말기의 잠금/열림 스케쥴에서 사용되는 시간 정보를 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

IsUsed :

UCSAPI_TERMINAL_TIMEZONE 구조체가 가지는 값에 대한 유효성 여부를 가진다.

StartHour / StartMin:

시작 시간 정보를 가진다.

EndHour / EndMin :

종료 시간 정보를 가진다.

UCSAPI_TERMINAL_DAY_SCHEDULE

Prototype:

```
typedef struct ucsapi_terminal_day_schedule
{
        UCSAPI_TERMINAL_TIMEZONE        Lock1;
        UCSAPI_TERMINAL_TIMEZONE        Lock2;
        UCSAPI_TERMINAL_TIMEZONE        Lock3;
        UCSAPI_TERMINAL_TIMEZONE        Open1;
        UCSAPI_TERMINAL_TIMEZONE        Open2;
        UCSAPI_ TERMINAL_TIMEZONE       Open3;
} UCSAPI_TERMINAL_DAY_SCHEDULE, * UCSAPI_TERMINAL_DAY_SCHEDULE_PTR;
```

Description:

요일 별 단말기의 잠금/열림 스케쥴 정보를 담는 구조체.

요일 별 스케줄은 하루에 가각 최대 세 개의 잠금/열림 시간대를 지정 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

Lock1 / Lock2 / Lock3:

하루중 단말기를 잠그기 위한 시간대 값을 가진다.

Open1 / Open2 / Open3:

하루중 단말기를 개방하기 위한 시간대 값을 가진다.

UCSAPI_HOLIDAY_TYPE

Prototype:

```
typedef UCSAPI_UINT8            UCSAPI_HOLIDAY_TYPE;

#define UCSAPI_HOLIDAY_TYPE_1       1
#define UCSAPI_HOLIDAY_TYPE_2       2
#define UCSAPI_HOLIDAY_TYPE_3       3
```

Description:

단말기의 잠금/열림 스케쥴에서 사용 될 휴일 타입을 지정. 휴일은 최대 3가지 타입을 지정 할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

UCSAPI_TERMINAL_HOLIDAY_INFO

Prototype:

```
typedef struct ucsapi_holiday_info
{
        UCSAPI_UINT8        Month;
        UCSAPI_UINT8        Day;
        UCSAPI_UINT8        HolidayType;
} UCSAPI_TERMINAL_HOLIDAY_INFO, * UCSAPI_TERMINAL_HOLIDAY_INFO_PTR
```

Description:

휴일 정보를 담는 구조체.

Month/Day:

휴일의 날짜 정보를 가진다.

Holidaytype:

휴일 타입 값을 가진다. UCSAPI_HOLIDAY_TYPE 참조

UCSAPI_TERMINAL_SCHEDULE

Prototype:

```
typedef struct ucsapi_terminal_schedule
{
        UCSAPI_TERMINAL_DAY_SCHEDULE        Sun;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Mon;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Tue;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Wed;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Thu;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Fri;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Sat;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Holiday1;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Holiday2;
        UCSAPI_TERMINAL_DAY_SCHEDULE        Holiday3;
        UCSAPI_TERMINAL_HOLIDAY_INFO        Holidays[100];
} UCSAPI_TERMINAL_SCHEDULE, * UCSAPI_TERMINAL_SCHEDULE_PTR;
```

Description:

단말기의 요일 별 잠금/열림 스케쥴 정보를 담은 구조체.
휴일은 최대 100개 까지 설정이 가능 하며, 그 타입은 Holiday1, Holiday2, Holiday3에 지정할 수 있다. 각각의 값들에 대한 설명은 다음과 같다.

Sun / Mon / Tue / Wed / Thu / Fri / Sat:

요일 별 잠금/열림 스케쥴 정보를 담은 구조체.

Holiday1 / Holiday2 / Holiday3:

휴일 타입 별 잠금/열림 스케쥴 정보를 담은 구조체.

Holidays:

휴일 정보를 담은 구조체. 각각의 휴일은 Holiday1, Holiday2, Holiday3 중 하나의 스케쥴에 적용을 받는다.

UCSAPI_SECURITY_LEVEL

Prototype:

```
typedef struct ucsapi_security_level
{
        UCSAPI_UINT8    Verify          :4;   /* 1:1 default level = 4*/
        UCSAPI_UINT8    Identify        :4;   /* 1:N default level = 5*/
} UCSAPI_SECURITY_LEVEL, * UCSAPI_SECURITY_LEVEL_PTR;
```

Description:

지문 인증 시 사용할 보안 레벨 정보를 담은 구조체. 이 값은 아래와 같은 값을 가질 수 있다.

1 - LOWEST
2 - LOWER
3 - LOW
4 - BELOW_NORMAL
5 - NORMAL
6 - ABOVE_NORMAL
7 - HIGH
8 - HIGHER
9 - HIGHEST

Verify:

1:1 인증 레벨 값. 기본 값으로 4를 가진다.

Identify

1:N 인증 레벨 값. 기본 값으로 5를 가진다.

UCSAPI_ANTIPASSBACK_LEVEL

Prototype:

```
typedef UCSAPI_UINT32           UCSAPI_ANTIPASSBACK_LEVEL;

#define UCSAPI_ANTIPASSBACK_LEVEL_NOT_USE                   0
#define UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_ALLOW  1
#define UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_PROHIBIT 2
```

Description:

단말기에 설정되어 있는 안티패스백의 레벨 값을 가진다. 단말기는 안티패스백 기능을 사용하기 위해 UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_ALLOW 또는 UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_PROHIBIT로 설정 하여야 한다.

UCSAPI_ANTIPASSBACK_LEVEL_NOT_USE:

안티패스백 사용안함.

UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_ALLOW:

서버와의 연결이 단절 상태에서 출입 허용.

UCSAPI_ANTIPASSBACK_LEVEL_WHEN_DISCONNECTION_PROHIBIT:

서버와의 연결이 단절 상태에서 출입 불가.

UCSAPI_NETWORK_INFO

Prototype:

```
typedef struct ucsapi_network_info
{
        UCSAPI_UINT8        NetworkType;
        UCSAPI_UINT8        IP[4];
        UCSAPI_UINT8        Subnet[4];
        UCSAPI_UINT8        Gateway[4];
} UCSAPI_NETWORK_INFO;
```

Description:

단말기의 네트워크 설정 정보를 담는 구조체.

NetworkType :

IP 주소의 타입 값을 가진다. 그 값이 0이면 고정 IP를 1이면 유동 IP를 지원한다.

IP :

단말기의 IP 주소 값을 담은 버퍼의 배열.

Subnet :

Subnet Mask 값을 담은 버퍼의 배열.

Gateway :
게이트웨이 주소 값을 담은 버퍼의 배열.

UCSAPI_SERVER_INFO

Prototype:

```
typedef struct ucsapi_server_info
{
        UCSAPI_UINT8            IP[4];
        UCSAPI_UINT16           Port;
        UCSAPI_UINT8            Reserved[2];
} UCSAPI_SERVER_INFO;
```

Description:

단말기가 서버로 접속하기 위한 네트워크 정보를 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

IP :
서버 IP의 주소 값이 담겨있는 버퍼 배열이다.

Port :
서버 접속을 위한 소켓 포트 값을 가진다.

UCSAPI_TERMINAL_OPTION_FLAG

Prototype:

```
typedef struct ucsapi_terminal_option_flag
{
        UCSAPI_UINT32       SecurityLevel       :1;
        UCSAPI_UINT32       InputIDLength       :1;
        UCSAPI_UINT32       AutoEnterKey        :1;
        UCSAPI_UINT32       Sound               :1;
        UCSAPI_UINT32       Authentication      :1;
        UCSAPI_UINT32       Application         :1;
```

```
    UCSAPI_UINT32       Antipassback        :1;
    UCSAPI_UINT32       Network             :1;
    UCSAPI_UINT32       Server              :1;
    UCSAPI_UINT32       InputIDType         :1;
    UCSAPI_UINT32       AccessLevel         :1;
    UCSAPI_UINT32       PrintText           :1;
    UCSAPI_UINT32       Schedule            :1;
} UCSAPI_TERMINAL_OPTION_FLAG, *UCSAPI_TERMINAL_OPTION_FLAG_PTR;
```

Description:

UCSAPI_TERMINAL_OPTION 구조체의 각 항목에 대한 참조 플래그 값을 가진다. 해당 항목의 플래그 값이 True 인경우에만 항목의 값을 참조 할 수 있다.
각 항목에 대한 설명은 UCSAPI_TERMINAL_OPTION 구조체의 설명을 참조.

UCSAPI_TERMINAL_OPTION

Prototype:

```
typedef struct ucsapi_terminal_option
{
    UCSAPI_TERMINAL_OPTION_FLAG  Flags;
    UCSAPI_SECURITY_LEVEL        SecurityLevel;
    UCSAPI_UINT8                 InputIDLength;
    UCSAPI_UINT8                 AutoEnterKey;
    UCSAPI_UINT8                 Sound;
    UCSAPI_UINT8                 Authentication;
    UCSAPI_UINT8                 Application;
    UCSAPI_UINT8                 Antipassback;
    UCSAPI_NETWOARK_INFO         Network;
    UCSAPI_SERVER_INFO           Server;
    UCSAPI_UINT8                 InputIDType;
    UCSAPI_UINT8                 AccessLevel;
    UCSAPI_UINT8                 PrintText[32];
    UCSAPI_TERMINAL_SCHEDULE     Schedule;
} UCSAPI_TERMINAL_OPTION, *UCSAPI_TERMINAL_OPTION_PTR;
```

Description:

단말기의 옵션 설정 값을 담는 구조체.
UCSAPI_SetOptionToTerminal / UCSAPI_GetOptionFromTerminal 함수에서 사용된다. 각각의 값들에 대한 설명은 다음과 같다.

Flags :
구조체의 각각의 항목에 대한 참조 플래그 값을 가진다.
UCSAPI_SetOptionToTerminal 함수를 사용하여 단말기의 옵션 항목을 설정하고자 한다면 해당 항목의 플래그 값을 True로 지정하고 항목 값을 지정한다.

SecurityLevel :
인증 시 사용할 보안 수준을 지정.
가질 수 있는 값은 UCBioBSP SDK의 UCBioAPI_FIR_SECURITY_LEVEL 정의 참조.

InputIDLength :
단말기에서 입력 ID의 길이 값을 가진다. UserID 사용시 최대 8을 지정 할 수있고 UniqueID 사용시 최대 20을 지정 할 수 있다.

AutoEnterKey :
단말기의 자동 엔터키 사용 여부의 값을 가진다. 이 기능은 InputIDLength 만큼 Key 입력이 있을 때 Enter Key를 자동으로 입력 시키는 기능이다.

Sound :
단말기의 사운드 볼륨 값을 갖는다. 볼륨 값은 0 ~ 20까지 지정 할 수 있다. 단말기의 사운드를 Mute로 설정 하고자 한다면 0을 지정한다.

Authentication :
단말기의 인증 방식 값을 가진다.
가질 수 있는 값은 1.5장 용어 설명의 "단말기 인증 방식" 참조.

Application:
단말기 프로그램의 모드 값을 가진다. 단말기는 그 기능에 따라 출입통제, 근태, 식수 등으로 사용 될수 있다. 가질 수 있는 값은 1.5장 용어 설명의 "단말기 프로그램 모드" 참조.

Antipassback:
단말기의 안티패스백 레벨 값을 가진다.

가질 수 있는 값은 UCSAPI_ANTIPASSBACK_LEVEL 정의 참조.

Network:
단말기의 네트워크 정보를 담은 구조체.

Server:
단말기가 서버로 접속하기 위한 네트워크 정보를 담은 구조체.

InputIDType:
단말기에서 인증 시 입력되는 ID의 타입 값을 가진다. 가질 수 있는 값은 다음과 같다.

0 – UserID
1 – UniqueID

AccessLevel:
접근 레벨 값을 가진다. 단말기에서 입력되는 인증 타입을 제한하는 기능으로 지정된 타입만을 인증 가능하게 한다. 가질 수 있는 값은 다음과 같다. 0을 기본 값으로 가진다.

0 – 제한 없음.
1 – 지문과 비밀번호 인증만 허용.

PrintText:
단말기에 연결된 식수 프린터기에 출력 할 문자열이 담겨있는 버퍼 배열이다.
이 값은 단말기에 식수 프린터기가 연결 되어 있는경우에 사용 할 수 있다.

Schedule : 잠금/개방 스케줄 정보를 담고 있는 구조체.

3.1.9 Monitoring related types

UCAPI_Type.h에 선언되어 있으며 모니터링 관련 Type들에 대해 정의 하고있다.

UCSAPI_TERMINAL_STATUS

Prototype:

typedef struct ucsapi_terminal_status

```
{
        UCSAPI_UINT32       Terminal;
        UCSAPI_UINT32       Door;
        UCSAPI_UINT32       Cover;
} UCSAPI_TERMINAL_STATUS, *UCSAPI_TERMINAL_STATUS_PTR;
```

Description:

단말기의 상태 값을 담은 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Terminal:

단말기의 잠금 상태 값을 가진다. 가질 수 있는 값은 다음과 같다.

0 – UnLock

1 – Lock

Door:

단말기의 잠금장치 상태 값을 가진다. 이 값은 모니터링 기능이 지원되는 잠금장치에서만 지원된다. 가질 수 있는 값은 다음과 같다.

0 – Close

1 – Open

2 – Not Use

Cover:

단말기의 뚜껑 상태 값을 가진다. 가질 수 있는 값은 다음과 같다.

0 – Close

1 – Open

3.3 API References

UCS SDK에서 사용되는 각종 API 대한 정의와 그 함수 사용법 및 인자들에 대해 설명한다.

3.3.1 Initialize API

UCS SDK를 시작/종료 하기 위한 API들에 대한 설명이다.

UCSAPI_ServerStart

Prototype:

```
UCSAPI_RETURN UCSAPI UCSAPI_ServerStart(
        IN UCSAPI_UINT32 MaxTerminal,
        IN UCSAPI_UINT32 Port,
        IN UCSAPI_INT32   Reserved,
        IN UCSAPI_CALLBACK_EVENT_HANDLER CallBackEventFunction);
```

Description:

UCSAPI 모듈을 초기화 하며, 서버 기능을 실행한다.

Parameters:

MaxTerminal::

최대 연결 단말기 수를 정의한다. SDK는 입력된 최대 단말기 수에 따라 내부 메모리 사용량을 미리 할당하여 속도를 개선할 수 있다. 단말기 연결이 최대 수를 넘을 경우 자동으로 메모리 사용량을 늘려 연결 수를 확보 한다.

Port:

단말기 접속을 위한 통신 포트. 서버는 지정된 포트로 단말기의 접속을 대기 한다. 기본 값은 9870 이다. 이 값을 변경 할경우에는 단말기의 포트 값도 함께 변경하여야 한다.

CallBackEventFunction:

응용 프로그램으로 이벤트 통지를 위한 콜 벡 함수에 대한 포인터

Returns:

UCSAPIERR_NONE

UCSAPIERR_FUNCTION_FAILED

Callback:

UCSAPI_CALLBACK_EVENT_CONNECTED

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_0 CallBack0

lParam:

UCSAPI_UINT8 TerminalIP[4]:

단말기 IP 주소를 가지는 버퍼의 배열

UCSAPI_ServerStop

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_ServerStop();

Description:

연결된 모든 단말기들의 접속을 해지하며, 서버 기능을 종료한다.

Parameters:

N/A.

Returns:

UCSAPIERR_NONE

Callback:

N/A

Callback Parameters:

N/A.

3.3.2 Terminal User Management API

단말기의 사용자를 관리 할 수 있는 API에 대해서 설명한다.

UCSAPI_AddUserToTerminal

Prototype:

```
UCSAPI_RETURN UCSAPI UCSAPI_AddUserToTerminal(
        IN UCSAPI_INT32    ClientID,
        IN UCSAPI_INT32    TerminalID,
        IN UCSAPI_BOOL     IsOverwrite,
        IN UCBioAPI_USER_DATA_PTR* pUserData);
```

Description:

지정한 단말기로 사용자 정보를 전송 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID

IsOverwrite:

이미 등록된 사용자인 경우 덮어쓰기 할지를 지정한다. 1을 기본 값으로 한다.

pUserData:

사용자 데이터를 담고 있는 구조체의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

UCSAPIERR_USER_NAME_SIZE
UCSAPIERR_UNIQUE_ID_SIZE
UCSAPIERR_INVALID_SUCURITY_LEVEL
UCSAPIERR_INVALID_PARAMETER
UCSAPIERR_CODE_SIZE
UCSAPIERR_PASSWORD_SIZE
UCSAPIERR_MAX_CARD_NUMBER
UCSAPIERR_MAX_FINGER_NUMBER
UCSAPIERR_PICTURE_SIZE

Callback:

UCSAPI_CALLBACK_EVENT_ADD_USER

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

UCSAPI_UINT32 UserID;

※ 주의사항
UCSAPI_AddUserToTerminal 함수를 이용하여 다수의 사용자 데이터를 단말기로 전송 할 경우 반드시 UCSAPI_AddUserToTerminal 호출후 UCSAPI_CALLBACK_EVENT_ADD_USER 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자 데이터를 전송 해야 한다.

UCSAPI_DeleteUserFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_DeleteUserFromTerminal(

IN UCSAPI_INT32 ClientID,

IN UCSAPI_INT32 TerminalID,

IN UCSAPI_INT32 UserID);

Description:

지정한 단말기로부터 사용자 데이터를 삭제 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

UserID:

삭제하고자 하는 사용자의 ID

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_DELETE_USER

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam :

UCSAPI_UINT32 UserID;

※ 주의사항
UCSAPI_DeleteUserFromTerminal 함수를 이용하여 단말기의 여러 사용자를 삭제 할 경우 반드시 UCSAPI_DeleteUserFromTerminal 호출 후 UCSAPI_CALLBACK_EVENT_DELETE_USER 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자를 삭제 해야 한다.

UCSAPI_DeleteAllUserFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_DeleteAllUserFromTerminal(

IN UCSAPI_INT32 ClientID,

IN UCSAPI_INT32 TerminalID);

Description:

지정한 단말기로부터 모든 사용자 데이터를 삭제 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_DELETE_ALL_USER

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

N/A

UCSAPI_GetUserCountFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetUserCountFromTerminal(

IN UCSAPI_INT32 ClientID,

IN UCSAPI_INT32 TerminalID);

Description:

지정한 단말기로부터 등록된 사용자의 개수를 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_GET_USER_COUNT

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

UCSAPI_UINT32 nUserCount

사용자의 개수를 담고 있다.

UCSAPI_GetUserInfoListFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetUserInfoListFromTerminal(
IN UCSAPI_INT32 ClientID,
IN UCSAPI_INT32 TerminalID);

Description:

지정한 단말기로부터 등록된 모든 사용자 정보 리스트를 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

Returns:

UCSAPIERR_NONE
UCSAPIERR_NOT_SERVER_ACTIVE
UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_GET_USER_INFO_LIST

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

UCSAPI_CALLBACK_PARAM_PTR_1 pCallback1
pCallback1.DataType = UCSAPI_CALLBACK_DATA_TYPE_USER_INFO

UCSAPI_GetUserDataFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetUserDataFromTerminal(

IN UCSAPI_INT32 ClientID,

IN UCSAPI_UINT32 TerminalID

IN UCSAPI_UINT32 UserID);

Description:

지정한 단말기로부터 사용자 데이터를 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

UserID:

사용자 ID.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_GET_USER_DATA

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam :

```
UCSAPI_CALLBACK_PARAM_PTR_1              pCallback1;
pCallback1.DataType = UCSAPI_CALLBACK_DATA_TYPE_USER_DATA
```

3.3.3 Log related API

단말기에 저장된 로그 데이터를 얻기 위한 API를 설명 한다.

UCSAPI_GetAccessLogCountFromTerminal

Prototype:

```
UCSAPI_RETURN UCSAPI UCSAPI_GetAccessLogCountFromTerminal(
        IN UCSAPI_INT32 ClientID,
        IN UCSAPI_INT32 TerminalID,
        IN UCSAPI_GET_LOG_TYPE LogType);
```

Description:

지정한 단말기로부터 인증 로그 개수를 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

LogType:

얻어 올 로그 타입을 지정한다. 가질수 있는 값은 UCSAPI_GET_LOG_TYPE 참조.

Returns:

UCSAPIERR_NONE
UCSAPIERR_NOT_SERVER_ACTIVE
UCSAPIERR_INVALID_PARAMETER
UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG_COUNT

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam :

UCSAPI_UINT32 nLogCount;

로그의 개수를 담고 있다.

UCSAPI_GetAccessLogFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetAccessLogFromTerminal(

IN UCSAPI_INT32 ClientID,

IN UCSAPI_INT32 TerminalID,

IN UCSAPI_GET_LOG_TYPE LogType);

Description:

지정한 단말기로부터 인증 로그 데이터를 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

LogType:

얻어 올 로그 타입을 지정한다. 가질수 있는 값은 UCSAPI_GET_LOG_TYPE 참조.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_PARAMETER

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

로그 Record가 여러 개 일경우 Record의 개수 만큼

UCSAPI_CALLBACK_EVENT_GET_ACCESS_LOG 이벤트가 발생한다. 총 로그의 개수와 현재 로그의 Index 정보는 pCallback0->Progress 값을 참조.

lParam:

UCSAPI_CALLBACK_PARAM_PTR_1 pCallback1;

pCallback1->DataType = UCSAPI_CALLBACK_DATA_TYPE_ACCESS_LOG;

3.3.4 Authentication related API

서버에서 인증을 수행하기 위한 API를 설명 한다.
단말기의 인증 방식이 N/S, NO 모드인 경우 단말기는 서버로 인증을 요청하게 된다.

UCSAPI_SendAuthInfoToTerminal

Prototype:

```
UCSAPI_RETURN UCSAPI UCSAPI_SendAuthInfoToTerminal(
        IN UCSAPI_INT32 TerminalID,
        IN UCSAPI_AUTH_INFO_PTR pUserAuthInfo);
```

Description:

단말기는 1:1 인증 시 사용자 인증 정보를 얻기 위해 응용 프로그램으로 UCSAPI_CALLBACK_EVENT_VERIFY_USER_AUTH_INFO 이벤트를 통지한다.
이때 응용 프로그램은 사용자의 인증 정보를 UCSAPI_AUTH_INFO 구조체에 담아 단말기로 즉시 전송 해야 한다.
UCSAPI_CALLBACK_EVENT_VERIFY_USER_AUTH_INFO 이벤트는 다음의 이벤트 보다 항상 먼저 발생한다.
UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1,
UCSAPI_CALLBACK_EVENT_VERIFY_CARD,
CSAPI_CALLBACK_EVENT_VERIFY_PASSWORD

Parameters:

TerminalID:
단말기 ID.

pUserAuthInfo:
사용자의 인증 정보를 담고있는 구조체의 포인터.

Returns:

UCSAPIERR_NONE
UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER
UCSAPIERR_INVALID_TERMINAL

Callback:
UCSAPI_CALLBACK_EVENT_VERIFY_USER_AUTH_INFO

Callback Parameters:
wParam:
UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:
UCSAPI_INPUT_ID_DATA_PTR pInputID;
단말기로부터 입력된 ID 정보를 담은 구조체.

UCSAPI_SendAuthResultToTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_SendAuthResultToTerminal(

IN UCSAPI_INT32 TerminalID,

IN UCSAPI_AUTH_RESULT_PTR pResult);

Description:

단말기는 사용자 인증을 위해 다음과 같은 이벤트를 응용 프로그램으로 통지한다.

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_N (1:N 지문 인증 요청)

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1 (1:1 지문 인증 요청)

UCSAPI_CALLBACK_EVENT_VERIFY_CARD (Card 인증 요청)

UCSAPI_CALLBACK_EVENT_VERIFY_PASSWORD (Password 인증 요청)

이때 응용 프로그램은 사용자 인증 결과를 UCSAPI_AUTH_RESULT 구조체에 담아 단말기로 즉시 전송 해야 한다.

Parameters:

TerminalID:

단말기 ID.

pResult:

인증 결과를 담은 구조체의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_N

UCSAPI_CALLBACK_EVENT_VERIFY_FINGER_1_TO_1

UCSAPI_CALLBACK_EVENT_VERIFY_CARD

UCSAPI_CALLBACK_EVENT_VERIFY_PASSWORD

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam :

UCSAPI_INPUT_DATA_PTR pInputData;

단말기로부터 입력된 샘플 데이터를 담은 구조체의 포인터.

3.3.5 Terminal Management API

단말기 관리를 위한 API를 설명한다.

UCSAPI_GetTerminalCount

Prototype:

```
UCSAPI_RETURN UCSAPI UCSAPI_GetTerminalCount(
        IN UCSAPI_UINT32* pTerminalCount);
```

Description:

서버에 접속 되어 있는 단말기의 개수를 얻어 온다.

Parameters:

pTerminalCount:

단말기의 개수를 담아올 값의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

Callback:

N/A

Callback Parameters:

N/A

UCSAPI_GetFirmwareVersionFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetFirmwareVersionFromTerminal(
IN UCSAPI_UINT32 ClientID,
IN UCSAPI_UINT32 TerminalID);

Description:

지정한 단말기의 펌웨어 버전을 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

Returns:

UCSAPIERR_NONE
UCSAPIERR_NOT_SERVER_ACTIVE
UCSAPIERR_INVALID_POINTER
UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_FW_VERSION

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

UCSAPI_DATA_PTR pVersion;
Version 정보를 담고있는 구조체의 포인터

UCSAPI_UpgradeFirmwareToTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_UpgradeFirmwareToTerminal(

IN UCSAPI_UINT32 ClientID,

IN UCSAPI_UINT32 TerminalID,

IN UCSAPI_CHAR_PTR pFilePath);

Parameters:

지정한 단말의 펌웨어를 업그레이드 한다. 업그레이드 진행 정보는 다음의 이벤트로 통지 된다.

UCSAPI_CALLBACK_EVENT_FW_UPGRADING / UCSAPI_CALLBACK_EVENT_FW_UPGRADED

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

pFilePath:

펌웨어 파일의 경로를 담은 값의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_FW_UPGRADING

UCSAPI_CALLBACK_EVENT_FW_UPGRADED

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_0 pCallback0;

업그레이드 진행 정보는 pCallback0->Progress 구조체 참조.

lParam:

N/A

UCSAPI_GetOptionFromTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_GetOptionFromTerminal(

IN UCSAPI_UINT16 ClientID,

IN UCSAPI_UINT32 TerminalID);

Description:

지정한 단말기로부터 옵션 설정 값을 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback Parameters:

UCSAPI_CALLBACK_EVENT_GET_TERMINAL_OPTION

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_0 pCallback0;

lParam:

UCSAPI_TERMINAL_OPTION_PTR pOption;

UCSAPI_SetOptionToTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_SetOptionToTerminal(

IN UCSAPI_UINT32 ClientID,

IN UCSAPI_UINT32 TerminalID,

IN UCSAPI_TERMINAL_OPTION_PTR pOption);

Description:

지정한 단말기의 옵션 값을 설정 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID.

pOption:

UCSAPI_TERMINAL_OPTION 구조체의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_SET_TERMINAL_OPTION

wParam

UCSAPI_CALLBACK_PARAM_PTR_0 pCallBack0;

lParam:

N/A

UCSAPI_OpenDoorToTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_OpenDoorToTerminal(

IN UCSAPI_UINT32 ClientID,

IN UCSAPI_UINT32 TerminalID);

Description:

지정한 단말기의 잠금 장치를 일시적으로 Open 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_OPEN_DOOR

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

lParam:

N/A

UCSAPI_SetAccessControlDataToTerminal

Prototype:

UCSAPI_RETURN UCSAPI UCSAPI_SetAccessControlDataToTerminal(

IN UCSAPI_UINT32 ClientID,

IN UCSAPI_UINT32 TerminalID,

IN UCSAPI_ACCESS_CONTROL_DATA_PTR pAccessControlData);

Description:

지정한 단말기로 출입제어 정보를 전송 한다. 시간대, 출입시간, 휴일, 출입그룹 정보는 각각 별도로 전송 해야 한다. 출입제어 정보는 단말기에서 인증 시 사용된다. 단말기는 저장된 출입제어 정보가 없는 경우 별도의 출입제어를 하지 않는다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID.

pAccessControlData:

출입제어 정보를 담은 구조체의 포인터.

Returns:

UCSAPIERR_NONE

UCSAPIERR_NOT_SERVER_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

Callback:

UCSAPI_CALLBACK_EVENT_SET_ACCESS_CONTROL_DATA

Callback Parameters:

wParam:

UCSAPI_CALLBACK_PARAM_PTR_0 pCallback0;

IParam:

N/A

4. API Reference for COM

이 장에서는 COM 모듈인 UCSAPICOM.dll을 사용하기 위한 Property 및 Method들에 대해 설명한다.

4.1 UCSAPI Object

UCSAPICOM.dll을 사용하기 위한 Main Object로서 UCS SDK의 기본 기능을 제공하고 각종 Child Object를 얻어오는 기본 Object로 동작한다. 때문에 SDK를 사용하기 위해서는 이 Object가 반드시 선언되어야 한다.

4.1.1 Properties

UCSAPI Object의 각종 Property에 대해 설명한다.

ErrorCode

Prototype:

[ReadOnly] long ErrorCode;

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.
Child Object의 Method 및 Property 설정 중 발생한 오류도 이 값으로 얻을 수 있다.

ConnectionsOfTerminal

Prototype:

[ReadOnly] long ConnectionsOfTerminal;

Description:

서버에 연결된 단말기의 개수에 대한 값이 담겨있다.

이 값은 GetTerminalCount() 메소드 호출 후 얻을 수 있다.

TerminalUserData

Prototype:

[ReadOnly] VARIANT TerminalUserData;

Description:

단말기로부터 사용자 정보를 얻기 위한 인터페이스를 얻는다.
GetUserInfoListFromTerminal / GetUserDataFromTerminal Method 호출 후 EventGetUserInfoList / EvetGetUserData 콜 백 이벤트 에서 사용자 정보를 얻기위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IterminalUserData 설명 참조.

ServerUserData

Prototype:

[ReadOnly] VARIANT ServerUserData;

Description:

사용자 정보를 단말기로 전송하기 위한 인터페이스를 얻는다.
AddUserToTerminal Method 호출 전 단말기로 전송하기 위한 사용자 정보를 설정하기위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IServerUserData 설명 참조.

AccessLogData

Prototype:

[ReadOnly] VARIANT AccessLogData;

Description:

단말기로부터 인증 로그를 얻기 위한 인터페이스를 얻는다.
GetAccessLog Method 호출 후 발생하는 EventGetAccessLog 콜 백 이벤트 에서 인증 로그를 얻기위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IAccessLogData 설명 참조.

AccessControlData

Prototype:

[ReadOnly] VARIANT AccessControlData;

Description:

단말기로 출입제어 데이터를 설정하기 위한 인터페이스를 얻는다.

SetAccessControlDataToTerminal Method 호출 전 출입권한 Data를 설정하기위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IAccessControlData 설명 참조.

4.1.2 Methods

UCSAPI Object의 각종 Method에 대해 설명한다.

ServerStart

Prototype:

HRESULT ServerStart(long MaxTerminal, long Port);

Description:

UCSAPI SDK을 초기화 하며, 서버 기능을 실행한다.

Parameters:

MaxTerminal:

최대 연결 단말기 수를 정의한다. SDK는 입력된 최대 단말기 수에 따라 내부 메모리 사용량을 미리 할당하여 속도를 개선할 수 있다. 단말기 연결이 최대 수를 넘을 경우 자동으로 메모리 사용량을 늘려 연결 수를 확보 한다.

Port :

단말기 접속을 위한 통신 포트. 기본 값으로 9870을 사용한다. 이 값을 변경 할경우에는 단말기의 포트 값도 함께 변경하여야 한다.

Related Properties

ErrorCode

Callback Event:

EventTerminalConnected(long TerminalID, BSTR TerminalIP);

Event Parameters:

TerminalID:

단말기 ID

TerminalIP:

단말기의 IP 주소 값을 담은 문자열.

ServerStop

Prototype:

HRESULT ServerStop();

Description:

연결된 모든 단말기들의 접속을 해지하며, 서버 기능을 종료한다.

Parameters:

N/A

Related Properties

ErrorCode

Callback Event:

N/A

Event Prameters

N/A

GetTerminalCount

Prototype:

HRESULT GetTerminalCount(void);

Description:

서버에 접속 되어 있는 단말기의 개수를 얻어 온다. ConnectionsOfTerminal Property를 이용해 얻을 수 있다.

Parameters:

N/A

Related Properties

ErrorCode, ConnectionsOfTerminal

Callback Event:

N/A

Event Prameters

N/A

GetFirmwareVersionFromTerminal

Prototype:

HRESULT GetFirmwareVersionFromTerminal(long ClientID, long TerminalID);

Description:

지정한 단말기의 펌웨어 버전을 얻어 온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

Related Properties

ErrorCode

Callback Event:

EventFirmwareVersion(long ClientID, long TerminalID, BSTR Version);

Event Prameters

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

Version:

펌웨어 버전 값을 문자열로 담고 있다.

UpgradeFirmwareToTerminal

Prototype:

HRESULT UpgradeFirmwareToTerminal (long ClientID, long TerminalID, BSTR FilePath);

Description:

지정한 단말기의 펌웨어를 업그레이드 한다. 업그레이드 진행 정보는 다음의 이벤트로 통지된다.
EventFirmwareUpgrading / EventFirmwareUpgraded

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

FilePath:

펌웨어 파일의 전체 경로를 문자열로 지정한다.

Related Properties

ErrorCode

Callback Event:

EventFirmwareUpgrading(long ClientID, long TerminalID, long CurrentIndex, long TotalNumber);
EventFirmwareUpgraded(long ClientID, long TerminalID);

Event Prameters

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

CurrentIndex:

현재 전송중인 데이터 Block의 Index 값을 담고 있다.

TotalNumber:

전송해야 할 전체 데이터 Block의 개수를 담고 있다.

OpenDoorToTerminal

Prototype:

HRESULE OpenDoorToTerminal(long ClientID, long TerminalID);

Description:

지정한 단말기의 잠금 장치를 일시적으로 Open 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID.

Related Properties

ErrorCode

Callback Event:

EventOpenDoor (long ClientID, long TerminalID);

Event Prameters

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerminalID:

단말기 ID

4.2 IServerUserData Interface

단말기로 사용자 정보를 전송하기 위한 인터페이스. 단말기로 사용자 정보를 전송하기위해 사용자 정보 관련 Properties을 설정 후 AddUserToTerminal Method를 호출한다. 사용자 정보를 전송후 EventAdduserToTerminal 이벤트의 에러코드를 확인하여 정상적으로 전송이 완료 되었는지 확인한다.

4.2.1 Properties

IServerUserData 인터페이스의 각종 Property에 대해 설명한다.

UserID

Prototype:

[WriteOnly] long UserID;

Description:

사용자 ID의 값을 지정한다. 이 값은 최대 8자리의 숫자 데이터만 지정 할 수 있다.

UniqueID

Prototype:

[WriteOnly] BSTR UniqueID;

Description:

고유 번호(사원번호) 값을 문자열로 지정한다. 이 값은 사용자 식별을 위해 UserID 대신 사용 할 수 있다. 최대 지정할 수 있는 데이터는 20bytes 까지다.

UserName

Prototype:

[WriteOnly] BSTR UserName;

Description:

사용자 이름 값을 문자열로 지정한다. 최대 지정할 수 있는 데이터는 16bytes 까지다.

AccessGroup

Prototype:

[WriteOnly] BSTR AccessGroup;

Description:

출입그룹 코드 값을 문자열로 지정한다. 코드 값은 고정 4bytes 문자열로 구성된다.

SecurityLevel

Prototype:

[WriteOnly] long SecurityLevel;

Description:

지문 인증을 위해 사용되어질 인증 보안 레벨을 설정 할 수 있다. 이 값은 아래와 같은 값을 가질 수 있다.

1 - LOWEST
2 - LOWER
3 - LOW
4 - BELOW_NORMAL
5 - NORMAL
6 - ABOVE_NORMAL
7 - HIGH
8 - HIGHER
9 - HIGHEST

기본값으로 1:1은 4의 값을 가지고 1:N은 5의 값을 가진다.

IsCheckSimilarFinger

Prototype:

[WriteOnly] BOOL IsCheckSimilarFinger;

Description:

사용자의 지문 데이터를 단말기로 추가 시 유사지문 체크 여부를 지정한다.
이 값을 True로 지정하는 경우 단말기는 등록 된 모든 사용자의 지문과 비교하여 유사지문이 존재하는지 검사 하여 유사지문이 검출 되면 등록 실패 처리한다. 이 Flag는 단말기로 사용자 추가작업을 느리게 함으로 지문 등록 사용자가 많은 경우 성능을 저하시키는 요인이 된다.

IsAdmin

Prototype:

[WriteOnly] BOOL IsAdmin;

Description:

사용자를 단말기 관리자로 지정 할 수 있다. 1명 이상의 관리자가 등록된 단말기는 설정 메뉴로 진입시 관리자 로그온 과정을 거쳐 단말기 메뉴 사용을 제한 할 수 있다.

IsIdentify

Prototype:

[WriteOnly] BOOL IsIdentify;

Description:

사용자를 1:N 지문 인증 가능 하도록 지정 할 수 있다.

Password

Prototype:

[WriteOnly] BSTR Password;

Description:

사용자가 비밀번호 인증 방식을 사용하는 경우 비밀번호 문자열을 지정 할 수있다. 최대 지정할 수 있는 데이터는 8bytes 까지다.

4.2.2 Methods

IServerUserData 인터페이스의 각종 Method에 대해 설명한다.

SetAuthType

Prototype:
HRESULT SetAuthType(BOOL AndOperation, BOOL Finger, BOOL FPCard, BOOL Password, BOOL Card, BOOL CardID);

Description:
사용자의 인증 타입을 지정 할 수 있다. 각각의 인증 타입은 AndOperation Flag에 따라 AND or OR 조합으로 사용이 가능하다.

Parameters:
AndOperation:
각각의 인증 타입을 AND or OR 조합으로 묶어 사용 하도록 지정 할 수 있다.
여기서 AND 조합인 경우 1, OR 조합의 경우는 0을 설정 한다. 자세한 사항은 1.5장 용어 설명의 사용자 속성을 참조.

Finger:
지문 인증을 가능하게 지정 할 수 있다.

FPCard:
지문 카드 인증을 가능하게 지정 할 수 있다. 지문 카드는 스마트 카드에 지문 정보를 저장하여 인증 하는 방식이다.

Password:
비밀번호 인증을 가능하게 지정 할 수 있다.

Card:
카드 인증을 가능하게 지정 할 수 있다.

CardID:

RFID를 UserID 또는 UniqueID 처럼 사용하도록 지정 할 수 있다. CardID는 카드의 RFID를 인증 수단으로 사용하지 않고, 단지 UserID 처럼 식별자로 사용하는 것을 말한다.
반드시 다른 인증 타입과 함께 AND 조합으로 지정 되어야 한다.

Related methods

AddUserToTerminal, SendAuthInfoToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetFPSampleData

Prototype:

HRESULT SetFPSampleData(BOOL bInitialize, long nSrcFPDataType, long nFPDataSize, VARIANT FPData1, VARIANT FPData2);

Description:

지문 인증을 위하여 변환된 FIR의 손가락별 Template의 Binary stream 데이터를 지정 할 수 있다. Template에 대한 자세한 설명은 UCBioBSP SDK 참조.

Parameters:

bInitialize:

FIR 데이터를 초기화 하고 새로 만들 것인지를 지정.
만약 이 값이 False이면 지금 추가하는 Template 데이터는 내부적으로 만들어진 FIR 데이터에 계속 추가되게 되어 다수개의 Template 데이터를 가지는 하나의 FIR 데이터가 만들어지게 된다. 이 값을 True로 하게 되면 기존에 있던 FIR을 모두 지우고 새로 만들게 된다.

nSrcFPDataType :

추가하고자 하는 Template의 Type 정보. 관련 값은 UCBioAPI_TEMPLATE_TYPE 참조.

nFPDataSize :

추가하고자 하는 Template의 크기 데이터.

FPData1:

추가하고자 하는 Template 데이터. (Binary stream 데이터)

FPData2:

추가하고자 하는 손가락의 두 번째 Template 데이터. (Binary stream 데이터)

Related properties:

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetCardData

Prototype:

HRESULT SetCardData(BOOL bInitialize, BSTR RFID);

Description:

Card 인증을 위하여 RFID 데이터를 지정 할 수 있다.

Parameters:

bInitialize:

RFID 데이터를 초기화 하고 새로 만들 것인지를 지정.
만약 이 값이 False이면 지금 추가하는 RFID 데이터는 내부적으로 만들어진 RFID 데이터에 계속 추가되게 되어 다수개의 RFID 데이터가 만들어지게 된다.
이 값을 True로 하게 되면 기존에 있던 RFID 데이터를 모두 지우고 새로 만들게 된다.

RFID :

추가하고자 하는 RFID 값을 문자열로 지정한다. 최대 지정할 수 있는 데이터는 16bytes 까지다.

Related properties:

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetPictureData

Prototype:

HRESULT SetPictureData(long PictureDataLength, BSTR PictureDataType, VARIANT PictureData);

Description:

사진 데이터를 Binary stream을 지정 할 수 있다.

Parameters:

PictureDataLength:

사진 데이터의 길이를 지정.

PictureDataType:

사진 데이터의 타입 값을 문자열로 지정한다. (현재는 "JPG" 포맷만 지원)
파일의 확장자 값을 문자열로 지정한다.

PictureData:

추가하고자 하는 사진 데이터. (Binary stream 데이터)

Related properties:

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetAccessDate

Prototype:

HRESULT SetAccessDate(long AccessDateType, long StartYear, long StartMonth, long StartDay, long EndYear, long EndMonth, long EndDay);

Description:

출입 가능 기간 또는 불 가능 기간을 지정 할 수 있다. 출입 기간 데이터는 사용안함, 출입 가능 기간, 출입 불가능 기간과 같이 총 3개의 데이터 Type을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

Parameters:

AccessDateType:

출입 기간 데이터의 타입을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

0 - 사용 하지 않음
1 - 인증 가능 기간을 지정
2 - 인증 불 가능 기간을 지정

StartYear / StartMonth / StartDay:

출입 기간의 시작 일을 지정한다.

EndYear / EndMonth / EndDay:

출입 기간의 종료 일을 지정한다.

Related properties:

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

AddUserToTerminal

Prototype:

HRESULT AddUserToTerminal(long ClinetID, long TerminalID, BOOL IsOverwrite);

Description:

지정한 단말기로 사용자 정보를 전송 한다 AddUserTerminal Method를 호출 하기전 사용자 정보 관련 Properties 및 Method를 사용하여 사용자 정보를 만들어야 한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

IsOverwrite:

이미 등록된 사용자인 경우 덮어쓰기 할지를 지정한다. 1을 기본 값으로 한다.

Related properties:

ErrorCode

Callback Event:

EventAddUser

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

※ 주의사항
AddUserToTerminal Mehtod를 이용하여 다수의 사용자 데이터를 단말기로 전송 할 경우 반드시

AddUserToTerminal Method 호출 후 EventAddUser 콜 백 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자 데이터를 전송 해야 한다.

4.3 ITerminalUserData Interface

단말기의 사용자 정보 관리 및 얻어오기 기능과 관련된 인터페이스. 사용자 개수, 사용자 정보 리스트, 사용자 데이터를 얻어 오거나 삭제하기 위해 이 인터페이스를 얻어와 사용해야 한다.

4.3.1 Properties

ITerminalUserData 인터페이스의 각종 Property에 대해 설명한다.

CurrentIndex / TotalNumber

Prototype:

```
[ReadOnly]    long    CurrentIndex;
[ReadOnly]    long    TotalNumber;
```

Description:

다수개의 사용자 정보 리스트를 얻는 경우 총 리스트의 개수와 현재 레코드의 Index를 담고 있다. GetUserInfoListFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal,

UserID

Prototype:

```
[ReadOnly]    long    UserID;
```

Description:

사용자 ID 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

UniqueID

Prototype:

[ReadOnly] BSTR UniqueID;

Description:

고유번호(사원번호) 값이 담겨있다.

GetUserInfoListFromTerminal, GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

UserName

Prototype:

[ReadOnly] BSTR UserName;

Description:

사용자 이름 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

AccessGroup

Prototype:

[ReadOnly] BSTR AccessGroup;

Description:

출입그룹 코드 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

IsAdmin

Prototype:

[ReadOnly] BOOL IsAdmin;

Description:

사용자의 관리자 여부 Flag 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

IsIdentify

Prototype:

[ReadOnly] BOOL IsIdentify;

Description:

1:N 지문 인증 가능 Flag 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsFinger, TotalFingerCount, FingerID, FPSampleDataLength, SampleNumber, FPSampleData, SecurityLevel

AccessDateType

Prototype:

[ReadOnly] long AccessDateType;

Description:

출입 기간 데이터의 타입 값을 담고있다. 가질 수 있는 값은 다음과 같다.

GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

0 - 사용 하지 않음
1 - 인증 가능 기간을 지정

2 - 인증 불 가능 기간을 지정

Related methods:

GetUserDataFromTerminal

Related properties:

StartAccessDate, EndAccessDate

StartAccessDate/EndAccessDate

Prototype:

[ReadOnly] BSTR StartAccessDate;

[ReadOnly] BSTR EndAccessDate;

Description:

출입 기간의 시작/종료 일을 문자열로 담고 있다.

데이터 포맷은 yyyy-MM-dd 형태이다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related Properties:

AccessDateType

SecurityLevel

Prototype:

[ReadOnly] long SecurityLevel;

Description:

지문 인증을 위해 사용되어질 인증 보안 레벨 값이 담겨있다.

GetUserDataFromTerminal Method 호출 후 얻을수 있다.

가질 수 있는 값은 다음과 같다.

1 - LOWEST

2 - LOWER

3 - LOW
4 - BELOW_NORMAL
5 - NORMAL
6 - ABOVE_NORMAL
7 - HIGH
8 - HIGHER
9 - HIGHEST

기본값으로 1:1은 4의 값을 가지고 1:N은 5의 값을 가진다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger, TotalFingerCount, FingerID, FPSampleDataLength, SampleNumber, FPSampleData

IsAndOperation

Prototype:

[ReadOnly] BOOL IsAndOperation;

Description:

인증 타입의 AND/OR 조합 사용 Flag 값이 담겨있다. 이 값은 Finger, Card, Password 인증 타입을 AND 또는 OR 조합으로 사용이 가능하도록 한다. 자세한 사항은 1.5장 용어 정리의 사용자 속성을 참조.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger, IsFPCard, IsCard, IsPassword

IsFinger

Prototype:

[ReadOnly] BOOL IsFinger;

Description:

사용자의 지문 인증 가능 Flag 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsAndOperation, TotalFingerCount, FingerID, FPSampleDataLength, SampleNumber, FPSampleData, SecurityLevel

IsFPCard

Prototype:

[ReadOnly] BOOL IsFPCard;

Description:

사용자의 지문카드 인증 가능 Flag 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsAndOperation

IsCard

Prototype:

[ReadOnly] BOOL IsCard;

Description:

사용자의 Card 인증 가능 Flag 값이 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsAndOperation, CardNumber, RFID

IsCardID

Prototype:

[ReadOnly] BOOL IsCardID;

Description:

Card 사용자의 RFID를 사용자 식별을 위한 ID로 사용할지의 Flag 값이 담겨있다.
CardID는 카드의 RFID를 인증 수단으로 사용하지 않고, 단지 UserID 처럼 식별자로 사용하는 것을 말한다. 반드시 다른 인증 타입과 함께 AND 조합으로 지정 되어야 한다..
GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsAndOperation

IsPassword

Prototype:

[ReadOnly] BOOL IsPassword;

Description:

비밀번호 인증 가능 Flag 값이 담겨있다.
GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserInfoListFromTerminal, GetUserDataFromTerminal

Related properties:

IsAndOperation, Password

Password

Prototype:

[ReadOnly] BSTR Password;

Description:

비밀번호 값이 문자열로 담겨있다. 최대 지정할 수 있는 데이터는 8bytes 까지다.
GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsPassword

CardNumber

Prototype:

[ReadOnly] long CardNumber;

Description:

사용자가 카드 인증 방식을 사용하는 경우 등록된 RFID의 개수 값이 담겨있다.
GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsCard, RFID

RFID

Prototype:

[ReadOnly] BSTR RFID(long nIndex);

Description:

사용자가 카드 인증 방식을 사용하는 경우 등록된 RFID의 데이터 값이 담겨있다.
GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Parameters:

nIndex

얻어올 RFID의 Index 번호:

Related methods:

GetUserDataFromTerminal

Related properties:

IsCard, CardNumber

PictureDataLength

Prototype:

[ReadOnly] long PictureDataLength;

Description:

사진 데이터의 크기 값이 담겨있다.

GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

PictureData

PictureData

Prototype:

[ReadOnly] VARIANT PictureData;

Description:

Binary stream 형태의 사진 데이터 값이 담겨있다. 데이터의 길이 값은 PictureDataLength Property에 담겨있다.

GetUserInfoListFromTerminal/GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

PictureDataLength

TotalFingerCount

Prototype:

[ReadOnly] long TotalFingerCount;

Description:

변환된 FIR의 총 손가락 개수를 담고 있다.

GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger, FingerID, FPSampleDataLength, SampleNumber, FPSampleData

FingerID

Prototype:

[ReadOnly] long FingerID(long nIndex);

Description:

변환된 FIR의 손가락 ID 정보를 배열로 담고 있다.

nIndex는 0부터 (TotalFingerCount-1)의 값을 가질 수 있다.

GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger,TotalFingerCount, FPSampleDataLength, SampleNumber, FPSampleData

SampleNumber

Prototype:

[ReadOnly] long SampleNumber;

Description:

변환된 FIR의 손가락별 Template의 개수를 담고 있다. 1또는 2의 값이 담기게 된다.
GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger, TotalFingerCount, FingerID, FPSampleDataLength, FPSampleData

FPSampleData

Prototype:

[ReadOnly] VARIANT FPSampleData(long nFigerID, long nSampleNum);

Description:

변환된 FIR의 손가락별 Template의 Binary stream 데이터를 담고있다.
nFingerID와 SampleNum은 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.
Binary stream 데이터의 길이 값은 FPSampleDataLength Property를 이용해 얻을 수 있다.
GetUserDataFromTerminal Method 호출 후 얻을 수 있다.

Parameters:

nFingerID:

얻어올 손가락 ID 번호

nSampleNum:

얻어올 Sample 번호. 0또는 1의 값을 사용한다.

Related methods:

GetUserDataFromTerminal

Related properties:

IsFinger, TotalFingerCount, FingerID, FPSampleDataLength, SampleNumber

4.3.2 Methods

ITerminalUserData 인터페이스의 각종 Method에 대해 설명한다.

GetUserCountFromTerminal

Prototype:

HRESULT GetUserCountFromTerminal(long ClinetID, long TerminalID);

Description:

지정한 단말기로부터 등록된 총 사용자의 개수를 얻어온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

Related Properties

ErrorCode

Callback Event:

EventGetUserCount

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

AdminNumber:

등록된 Admin의 개수.

UserNumber:

등록된 사용자의 개수

GetUserDataFromTerminal

Prototype:

HRESULT GetUserDataFromTerminal(long ClinetID, long TerminalID, long UserID);

Description:

지정한 단말기로부터 사용자 데이터를 얻어온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

UserID:

사용자 ID.

Related Properties

ErrorCode, UserID, UserName, UniqueID, AccessGroup, AccessDateType, StartAccessDate, EndAccessDate, IsAdmin, IsIdentify, IsAndOperation, IsFinger, IsFPCard, IsCard, IsPassword, IsCardID, Passowrd, CardNumber, RFID, SecurityLevel, TotalFingerCount, FingerID, FPSampleDataLength, SampleNumber, FPSampleData, PictureDataLength, PictureData

Callback Event:

EventGetUserData

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

DeleteUserFromTerminal

Prototype:

HRESULT DeleteUserFromTerminal(long ClinetID, long TerminalID, long UserID);

Description:

지정한 단말기로부터 사용자 데이터를 삭제한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

UserID:

사용자 ID.

Related Properties

ErrorCode

Callback Event:

EventDeleteUser

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

UserID:

사용자 ID.

DeleteAllUserFromTerminal

Prototype:

HRESULT DeleteAllUserFromTerminal(long ClinetID, long TerminalID);

Description:

지정한 단말기로부터 모든 사용자 데이터를 삭제한다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

Related Properties

ErrorCode

Callback Event:

EventDeleteAllUser

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

4.4 IAccessLogData Interface

단말기의 인증 로그 얻어오기 기능과 관련된 인터페이스. 단말기의 인증 로그를 얻어오기 위해 이 인터페이스를 얻어와 사용해야 한다.

4.4.1 Properties

IAccessLogData 인터페이스의 각종 Property에 대해 설명한다.

CurrentIndex / TotalNumber

Prototype:

```
[ReadOnly]    long    CurrentIndex;
[ReadOnly]    long    TotalNumber;
```

Description:

다수개의 로그 레코드를 얻는 경우 총 레코드 개수와 현재 레코드의 Index를 담고 있다. GetAccessLogCountFromTerminal/GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

UserID

Prototype:

```
[ReadOnly]    long    UserID;
```

Description:

사용자 ID 값이 담겨있다.
GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

AuthType

Prototype:

[ReadOnly] long AuthType;

Description:

인증 타입 값을 담고 있다. 가질 수 있는 값은 다음과 같다.

0 - 1:N 지문 인증
1 - 1:1 지문 인증
2 - 지문 Card 인증
3 - Card 인증
4 - 비밀번호 인증

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

AuthMode

Prototype:

[ReadOnly] long AuthMode;

Description:

인증 Mode 값을 담고 있다. 가질 수 있는 값은 다음과 같다.

0 - 출근
1 - 퇴근
2 - 일반 (일반적인 입출입)
3 - 외근
4 - 복귀

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

DateTime

Prototype:

[ReadOnly] BSTR DateTime;

Description:

인증 시각 데이터를 문자열로 담고 있다.

데이터 Format은 "yyyy-MM-dd hh:mm:ss" 형태이다.

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

IsAuthorized

Prototype:

[ReadOnly] BOOL IsAuthorized;

Description:

인증 결과 값이 담겨있다. 이 값이 인증 성공 0의 값을 가지고 실패이면 1의 값을 가진다.

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

RFID

Prototype:

[ReadOnly] BSTR RFID;

Description:

인증 타입이 Card인 경우 RFID 값이 문자열로 담겨있다.

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

PictureDataLength

Prototype:

[ReadOnly] long PictureDataLength;

Description:

사진 데이터의 크기 값을 담고 있다.

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

Related properties:

PictureData

PictureData

Prototype:

[ReadOnly] VARIANT PictureData;

Description:

Binary stream 형태의 사진 데이터를 담고 있다. 데이터의 길이 값은 PictureDataLength Property에 담겨있다.

GetAccessLogFromTerminal Method 호출 후 얻을 수 있다.

Related methods:

GetAccessLogFromTerminal

Related properties:

PictureDataLength

4.4.2 Methods

IAccessLogData 인터페이스의 각종 Method에 대해 설명한다.

GetAccessLogCountFromTerminal

Prototype:

HRESULT GetAccessLogCountFromTerminal(long ClinetID, long TerminalID, long LogType);

Description:

지정한 단말기에 인증 로그의 개수를 얻어온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

LogType:

얻어올 로그의 타입을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

0 - 새로운 로그
1 - 이미 서버로 전송한 로그
2 - 저장된 모든 로그

Related properties

ErrorCode

Callback Event:

EventGetAccessLogCount

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

LogCount:

인증 로그의 개수.

GetAccessLogFromTerminal

Prototype:

HRESULT GetAccessLogFromTerminal(long ClinetID, long TerminalID, long LogType);

Description:

지정한 단말기에 저장된 인증 로그를 얻어온다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

LogType:

얻어올 로그의 타입을 지정할 수 있다. 가질 수 있는 값은 다음과 같다.

0 - 새로운 로그
1 - 이미 서버로 전송한 로그
2 - 저장된 모든 로그

Related Properties

ErrorCode, TotalNumber, CurrentIndex, UserID, DateTime, AuthType, AuthMode, IsAuthorized, RFID, PictureDataLength, PictureData

Callback Event:

EventGetAccessLog

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

4.5 IAccessControlData Interface

출입 제어 데이터를 단말기로 전송하기 위한 인터페이스. 단말기로 출입 제어 정보를 설정하기 위해 이 인터페이스를 얻어와 사용해야 한다.

4.5.1 Properties

IAccessControlData 인터페이스의 각종 Property에 대해 설명한다.

4.5.2 Methods

IAccessControlData 인터페이스의 각종 Method에 대해 설명한다.

InitData

Prototype:
HRESULT InitData(void);

Description:
출입 제어 데이터를 초기화 하고 새로 만들 것인지를 지정.

Parameters:
N/A

Related properties
ErrorCode

Callback Event:
N/A

Event Parameters:
N/A.

SetTimeZone

Prototype:

HRESULT SetTimeZone(BSTR Code, long nIndex, long StartHour, long StartMinute, long EndHour, long EndMinute);

Description:

하루 중 인증 가능한 시간대를 추가한다.
추가 가능한 시간대 코드는 최대 128개 이며, 하나의 시간대 코드에 추가 가능한 시간대는 최대 12개 이다, SDK는 추가된 정보를 배열로 담고 있다.

Parameters:

Code:

설정할 시간대의 식별자 코드 값으로 고정 4바이트 문자열.

nIndex:

시간대 정보에 대한 Index. 이 값은 0부터 11의 값을 가질 수 있다.

StartHour / StartMinute:

시간대의 시작 시간을 지정 할 수 있다.

EndHour / EndMinute:

시간대의 종료 시간을 지정 할 수 있다.

Related methods

SetAccessControlDataToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetAccessTime

Prototype:

HRESULT SetAccessTime(BSTR Code, BSTR Sun, BSTR Mon, BSTR Tue, BSTR Wed, BSTR Thu, BSTR, Fri, BSTR Sat, BSTR Hol, BSTR Holiday);

Description:

요일 별 출입가능시간을 추가한다.
추가 가능한 출입가능시간 코드는 최대 최대 128개 이며, SDK는 추가된 정보를 배열로 담고 있다.

Parameters:

Code:

설정할 출입가능시간의 식별자 코드 값으로 고정 4바이트 문자열.

Sun/Mon/Tue/Wed/Thu/Fri/Sat/Hol:

인증 시 사용 할 요일 별 출입가능시간대 코드와 휴일에 적용될 출입가능시간대 코드를 지정 할 수 있다.

Holiday:

SetHoliday Method에서 설정한 휴일 코드를 지정 할 수 있다. 지정한 휴일 코드는 Hol 코드의 시간대가 적용된다.

Related methods

SetAccessControlDataToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetHoliday

Prototype:

HRESULT SetHoliday(BSTR Code, long nIndex, long Month, long Day);

Description:

휴일 정보를 추가한다.

추가 가능한 휴일 코드는 최대 64개이며, 하나의 휴일 코드에 추가 가능한 휴일은 최대 32개 이다. SDK는 추가된 정보를 배열로 담고 있다.

Parameters:

Code:

설정할 휴일의 식별자 코드 값으로 고정 4바이트 문자열.

nIndex:

추가될 휴일의 Index 값. 이 값은 0부터 63의 값을 가질 수 있다.

Month / Day:

휴일에 대한 날짜을 지정 할 수 있다.

Related methods

SetAccessControlDataToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetAccessGroup

Prototype:

HRESULT SetAccessGroup(BSTR Code, long nIndex, BSTR AccessTime);

Description:

출입그룹 정보를 추가한다.

추가 가능한 출입그룹 코드는 최대 128개이며, 하나의 출입그룹 코드에 추가 가능한 출입가능시간 코드는 최대 4개 이다, SDK는 추가된 정보를 배열로 담고 있다.

Parameters:

Code:

설정할 출입그룹의 식별자 코드 값으로 고정 4바이트 문자열.

nIndex:

추가될 출입가능시간 코드의 Index 값. 이 값은 0부터 3의 값을 가질 수 있다.

AccessTime:

출입그룹에서 사용할 출입가능시간 코드를 지정 할 수 있다.

Related methods

SetAccessControlDataToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SetAccessControlDataToTerminal

Prototype:

HRESULT SetAccessControlDataToTerminal(long ClientID, long TerminalID, long DataType);

Description:

지정한 단말기로 SetTimeZone, SetAccessTime, SetHoliday, SetAccessTime, SetAccessGroup Method에 의해 추가된 출입제어 데이터를 단말기로 전송한다. 시간대, 출입시간, 휴일, 출입그룹 정보는 각각 별도로 전송 해야 한다. 출입제어 정보는 단말기에서 인증 시 사용된다. 단말기는 저장된 출입제어 정보가 없는 경우 별도의 출입제어를 하지 않는다.

Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.).

TerminalID:

단말기 ID.

DataType:

단말기로 전송 할 출입제어 데이터를 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

0 - 시간대 정보
1 - 휴일 정보
2 - 출입가능시간 정보
3 - 출입그룹 정보

Related methods

SetTimeZone, SetHoliday, SetAccessTime, SetAccessGroup

Related properties

ErrorCode

Callback Event:

HRESULT EventSetAccessControlData(long ClientID, long TerminalID, long DataType);

Event Parameters:

ClientID:

작업을 요청한 클라이언트의 ID. (Client / Server 모델 개발 시에 사용 된다.)

TerinalID:

단말기 ID.

DataType:

출입제어 데이터의 타입 값.

4.6 IServerAuthentication Interface

서버 인증을 수행하기 위한 인터페이스. 응용 프로그램은 단말기로부터 사용자 인증 요청에 대하여 응답하기 위해 이 인터페이스를 얻어와 사용해야 한다.

4.6.1 Properties

IServerAuthentication 인터페이스의 각종 Property에 대해 설명한다.

4.6.2 Methods

IServerAuthentication 인터페이스의 각종 Method에 대해 설명한다.

SetAuthType

Prototype:
HRESULT SetAuthType(BOOL AndOperation, BOOL Finger, BOOL FPCard, BOOL Password, BOOL Card, BOOL CardID);

Description:
사용자의 인증 타입을 지정 할 수 있다. 각각의 인증 타입은 AndOperation Flag에 따라 AND or OR 조합으로 사용이 가능하다. 반드시 SendAuthInfoToTerminal Method 호출 전 사용하여야 한다.

Parameters:
AndOperation:
이 값이 True인 경우 각각의 인증 타입은 AND 조합으로 인증이 가능하며 False 인 경우는 OR 조합으로 인증이 가능하다.

Finger:
지문 인증을 가능하게 지정 할 수 있다.

FPCard:
지문 카드 인증을 가능하게 지정 할 수 있다. 지문 카드는 스마트 카드에 지문 정보를 저장하여 인증 하는 방식이다.

Password:
비밀번호 인증을 가능하게 지정 할 수 있다.

Card:
카드 인증을 가능하게 지정 할 수 있다.

CardID:

Card 번호를 ID로 사용 가능하도록 지정 할 수 있다. RFID를 UserID 또는 UniqueID 처럼 사용하도록 지정 할 수 있다. CardID는 카드의 RFID를 인증 수단으로 사용하지 않고, 단지 UserID 처럼 식별자로 사용하는 것을 말한다. 반드시 다른 인증 타입과 함께 AND 조합으로 지정 되어야 한다.

Related methods

SendAuthInfoToTerminal

Related properties

ErrorCode

Callback Event:

N/A

Event Parameters:

N/A

SendAuthInfoToTerminal

Prototype:

HRESULT SendAuthInfoToTerminal(long TerminalID, long UserID, BOOL IsAccessibility, long ErrorCode);

Description:

단말기가 서버 인증 모드로 동작시 응용 프로그램은 EventAuthTypeWithUserID, EventAuthTypeWithUniqueID 이벤트에 대한 응답으로 사용자의 인증 정보를 단말기로 즉시 전송 하여야 한다. 반드시 SetAuthType Method 호출 후 사용하여야 한다.

Parameters:

TerminalID:

단말기 ID.

UserID:

인증 시도한 사용자의 ID.

IsAccessibility:

사용자가 출입 가능자인지 에대한 Flag 값을 지정한다. 이 값이 False인 경우 단말기는 인증 실패 처리 한다.

Related Properties

ErrorCode

Callback Event:

N/A

Event Prameters

N/A

SendAuthResultToTerminal

Prototype:

HRESULT SendAuthResultToTerminal(long TerminalID, long UserID, BOOL IsAccessibility, BOOL IsVisitor, BOOL IsAuthorized, BSTR AuthorizedTime, long ErrorCode);

Description:

단말기는 사용자 인증을 위해 다음과 같은 이벤트를 응용 프로그램으로 통지한다. EventVerifyCard, EventVerifyPassword, EventVerifyFinger_1_TO_1, EventVerifyFinger_1_TO_N 이때 응용 프로그램은 사용자의 인증 결과를 단말기로 즉시 전송해야 한다.

Parameters:

TerminalID:

단말기 ID.

UserID:

인증 되거나 인증 시도한 사용자의 ID.

IsAccessibility:

사용자가 출입 가능자인지에 대한 Flag를 지정한다. 이 값이 False인 경우 단말기는 인증 실패 처리 한다.

IsVistor:

방문자 여부를 지정한다.

IsAuthorized:

인증 성공 여부를 지정한다.

AuthorizedTime:

인증 시각을 지정한다. 이 값은 "yyyy-MM-dd hh:mm:ss" 형태의 문자열을 지정한다.

ErrorCode:

인증시 발생하는 에러코드를 지정 할 수 있다. 이 값이 0 이면 에러 없음. 자세한 것은 ErrorCode표 참조.

Related Properties

ErrorCode

Callback Event:

N/A

Event Prameters

N/A