UNION COMMUNITY
Server SDK For Windows

Programmer's Guide
Version 3.0

June, 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. 개요

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

UCS SDK는 UCB(UNION COMMUNITY Biometric) SDK와 함께 지문인식 서버 응용 프로그램 인터페이스(Application Programming Interface, API)를 제공하기 위한 것으로 사용자 등록 및 인증을 위한 Biometric API 와 네트워크형 단말기와 통신 할 수 있는 서버 API로 구성 되어 있다.

1.1 적용

UCS SDK는 UCB SDK와 함께 지문 인식 기술에 대한 응용 프로그램 인터페이스(Application Programming
Interface)와 네트워크형 단말기 제품과 연동 할 수 있는 서버 응용 프로그램 인터페이스(Server Application Programming Interface)를 정의 하고 있다. 따라서 지문인증시스템 개발 시에 적용하여, 다양한 지문인식에 대한 융합과 등록, 인증 및 인식의 수행 시에 최적의 기능을 갖도록 하는데 활용할 수 있다.

1.2 특징

■ 중앙 집중 관리 방식

단말기가 UCS 모듈에 접속 하는 방식으로 중앙에서 모든 단말기들을 집중 관리 할 수 있으며, 이런 한 방식은 공중망을 이용한 네트워크 구성에서 많은 이점을 가지고 있다.

■ 단말기 관리를 위한 다양한 API 제공

UCS SDK는 단말기의 다양한 기능들을 제어하기 위한 모든 API를 제공 한다.

■ COM 모듈 제공

UCS SDK에서는 C/C++ 개발자 뿐만 아니라 Visual Basic 이나 Delphi등을 사용 하는 개발자를 위하여 이들 툴에서 개발을 쉽게 할 수 있도록 COM 기반의 모듈을 함께 제공한다.

■ 다양한 인증 방식 제공

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

2. SDK 구성

2.1 개발 환경

UCS(UNION COMMUNITY Server) SDK에서 제공되는 모든 모듈은 VC++ 6.0에서 컴파일 되어졌으며, Visual C++ 등의 대부분의 32bit 컴파일러에서는 이 SDK를 사용하여 프로그래밍이 가능하다.

UCS SDK에서는 C/C++ 개발자 뿐만 아니라 Visual Basic 이나 Delphi등을 사용 하는 개발자를 위하여 이들 툴에서 개발을 쉽게 할 수 있도록 COM 모듈을 함께 제공한다.

2.3 모듈 구성

■ Basic 모듈 : UCSAPI.dll

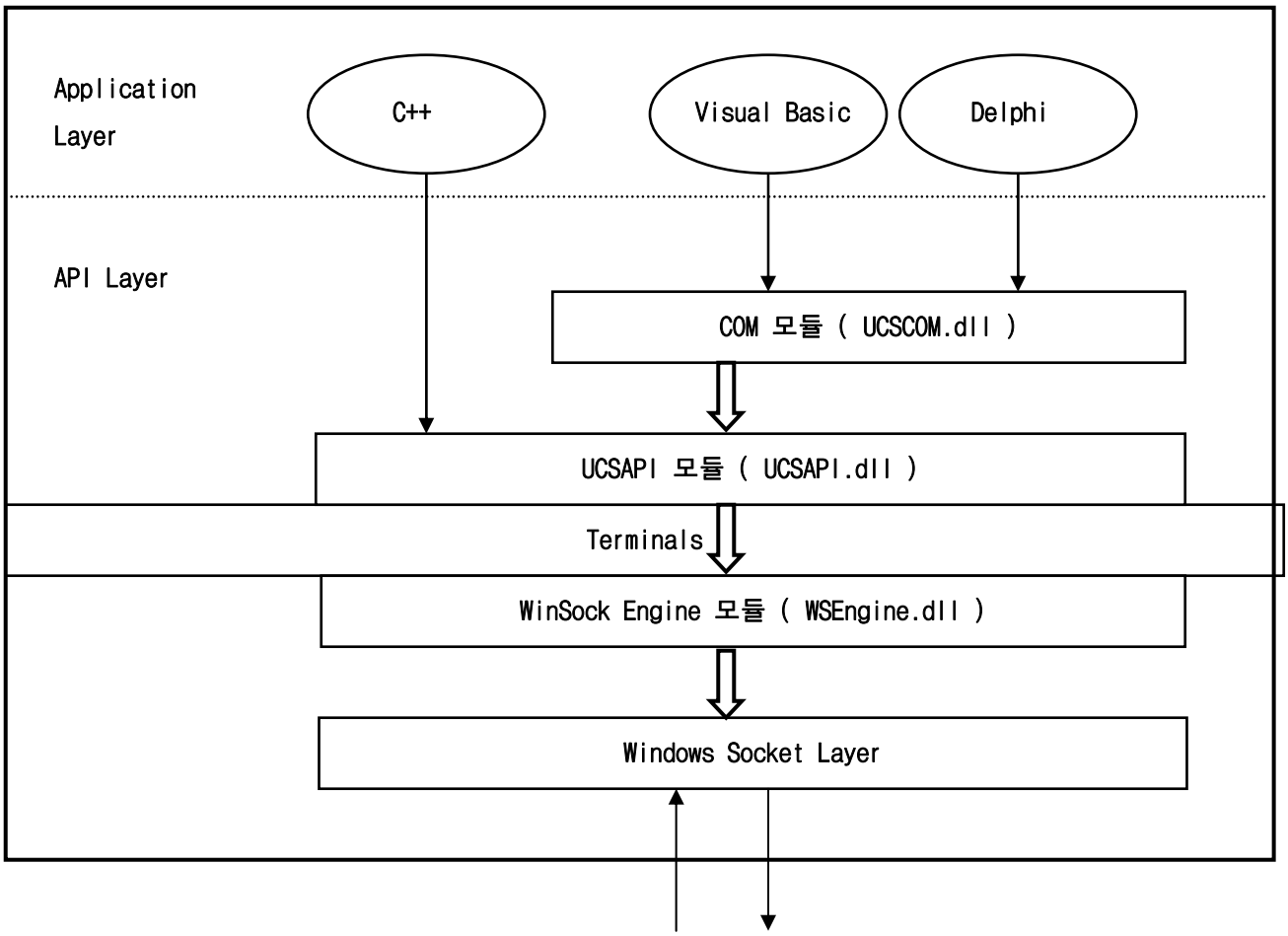
UCSAPI.dll은 네트워크형 지문인식 단말기와 통신하기 위한 기능을 구현하고 있는 기본 모듈이다.

■ COM 모듈 : UCSCOM.dll

UCSCOM 모듈은 Visual Basic 이나 Delphi등 RAD Tool 개발자를 위하여 개발 되었다.
UCSCOM 모듈은 UCSAPI 모듈을 바탕으로 작성 되었으며 UCSAPI 모듈의 모든 기능을 제공하지는 않는다.

■ Winsock Engine 모듈 : WSEngine.dll

WSEngine.dll은 Windows Socket IO(Input/Output)를 담당하는 모듈이며, 서버의 용량 및 성능에 따라
교체 되어질 수 있다.



2.4 용어 설명

■ UCSAPI (UNION COMMUNITY Server API)

㈜유니온커뮤니티에서 제공하는 네트워크형 지문인식 단말기와 통신 할 수 있는 기능을 제공하는 API 모듈을 말한다.

■ UCSCOM (UNION COMMUNITY Server COM)

UCSAPI모듈을 바탕으로 하며 Visual Basic 이나 Delphi등 RAD Tool 개발자를 위하여 개발 되어진 모듈을 말한다.

■ 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 모델의 응용 프로그램을 개발하고자 할 경우 사용 된다.
서버 모듈은 멀티 클라이언트 환경을 지원하기 위해 각각의 클라이언트들을 구분하기 위한 Key가 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 일대다수의 처리 과정이다.

■ 인증 요청 타입 (Authentication Request Type)

지문인식 단말기는 사용자 식별을 위하여 서버로 인증 요청 시 다양한 인증 타입을 사용 할 수 있다.

요청 타입	값(상수)	내용

1:N 지문 인증	0	1:N 지문 인증
1:1 지문 인증	1	1:1 지문 인증
지문 카드 인증	2	스마트 카드에 지문 정보를 저장하여 입력 지문과 저장 지문간 1:1 인증을 수행 (사용 하지 않음)
카드 인증	3	카드 인증
패스워드 인증	4	패스워드 인증

■ 인증 타입 (Authentication Type)

지문인식 단말기는 사용자 식별을 위하여 다양한 인증 타입을 지원 한다.

타입	값(상수)	내용
지문	0	인증 수단으로 지문을 사용 한다.
지문카드	1	인증 수단으로 지문 카드를 사용 한다.
비밀번호	2	인증 수단으로 비밀번호를 사용 한다.
카드	3	인증 수단으로 카드를 사용 한다.
카드 또는 지문	4	인증 수단으로 카드 또는 지문을 사용 한다.
카드 와 지문	5	인증 수단으로 카드와 지문의 조합을 사용 한다.
카드 또는 비밀번호	6	인증 수단으로 카드 또는 비밀번호를 사용 한다.
카드 와 비밀번호	7	인증 수단으로 카드와 지문의 조합을 사용한다.
(ID 와 지문) 또는 (카드와 지문)	8	인증 수단으로 ID와 지문의 조합 또는 카드와 지문의 조합을 사용한다.
(ID 와 비밀번호) 또는 (카드와 비밀번호)	9	인증 수단으로 ID와 비밀번호의 조합 또는 카드와 비밀번호의 조합을 사용한다.
지문 과 비밀번호	10	인증 수단으로 지문과 비밀번호의 조합을 사용한다.
지문 또는 비밀번호	11	인증 수단으로 지문 인증 실패 후 비밀번호를 사용한다.

■ 지문 인증 레벨 (Securiy Level)

성공적인 인증을 위한 레벨 값 이며, 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.

인증레벨	FMR 값	설명
1	1400 (~ 1800)	최저 레벨

2	1800 (~ 2200)	
3	2200 (~ 2600)	
4	2600 (~ 3000)	1:1 인증 시 권장 레벨
5	3000 (~ 3500)	1:N 인증 시 권장 레벨
6	3500 (~ 4000)	
7	4000 (~ 4500)	
8	4500 (~ 5000)	
9	5000 (~ 9999)	최고 레벨

■ 단말기 동작 모드 (Working Mode)

사용자의 인증 및 로그 기록에 대한 동작 방법을 정의 한다.

모드	값(상수)	Content
N / S	0	기본적으로 서버에서 사용자 인증을 수행 하며, 네트웍 단절 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	기본적으로 단말기에서 사용자 인증을 수행 하며, 단말기에 저장되어 있지 않은 사용자 인증 요구에 대해서는 서버로 인증 요청 한다. 인증기록은 항시 단말기에 저장되며, 서버와 네트웍 연결 시 에는 인증 결과 만을 서버로 전송하여 저장 하게 한다.
NO	2	서버에서 사용자 인증 만을 수행 한다.
SO	3	단말기에서 사용자 인증 만을 수행 한다.

■ 단말기 작업 모드 (Operation Mode)

단말기에서 지원되는 운영 방법을 정의 한다.

모드	값(상수)	내용
출입통제	0	단말기를 출입 통제 모드로 운영 한다.
근태 모드	1	단말기를 근태 관리 모드로 운영 한다.
식수 모드	2	단말기를 식수 관리 모드로 운영 한다.

■ 인증 모드 (Authentication Mode)

사용자 인증 시 인증에 대한 목적을 정의 한다.

모드	값(상수)	내용
출근	0	출근 인증 모드
퇴근	1	퇴근 인증 모드
일반	2	일반 인증 모드
외근	3	외근 인증 모드
복귀	4	복귀 인증 모드

■ 로그 타입 (Log Type)

UCS SDK는 단말기로부터 로그 데이터를 얻어 오기 위하여 세가지 타입의 로그를 정의 한다.

타입	값(상수)	내용

서버로 미 전송된 로그	0	서버로 전송 되지 않은 새로운 로그
이미 서버로 전송된 로그	1	서버로 이미 전송 완료된 로그
모든 로그	2	단말기에 저장되어 있는 모든 로그

■ 사용자 속성 (User Flag)

사용자의 관리자 여부 및 인증 타입을 정의하는 1 바이트 크기의 데이터 필드 이다.

7 Bit	6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit	0 Bit
Admin	Identify	Operation & or |	Card ID	Card	패스워드	음성	지문

사용자 속성은 다음의 12가지 값으로 표현 되어 질 수 있다.

① 지문(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 지문카드

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 비밀번호

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 카드

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 카드 또는 지문

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 카드 와 지문

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 카드 또는 비밀번호

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 카드 와 비밀번호

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 와 지문) 또는 (카드 와 지문)

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 와 비밀번호) or (카드 와 비밀번호)

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 지문 과 비밀번호

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 지문 또는 비밀번호 : 지문인증 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

■ 단말기 관리자 (Terminal Admin)

단말기 관리자는 단말기의 설정 정보를 변경 하거나 사용자를 등록/삭제 할 수 있는 권한을 가진 사용자를 말한다.

■ 단말기 상태

단말기는 자신의 상태 및 단말기 주변에 부착된 장치의 상태 정보를 주기적 또는 상태 변화가 있을 경우 즉시 서버로 그 상태를 전송 한다.

-단말기 상태 : 이 값은 단말기의 “잠금 스케줄링” 기능에 의하여 변경되는 상태 값 이다.

-잠금 장치 상태 : 이 값은 단말기의 “개방 스케줄링” 기능에 의하여 변경되는 상태 값 이다.

-도어 모니터링 상태 : 이 값은 모니터링이 지원되는 잠금 장치로부터 수신되는 도어의 상태 값이다.

-단말기 커버 상태 : 이 값은 단말기 커버의 상태를 반환하는 값이다.

상태	값(상수)	설명
단말기 상태	0	잠금 해제 상태

	1	잠금 상태 (단말의 잠금 상태에서는 서버와의 연결은 유지하지만, 서버로부터 가 아닌 단말에 대한 출입을 불가능하게 한다. 단, 관리자 출입허용 옵션을 사용 할 경우에는 관리자 만이 출입 할 수 있게 된다.)
잠금 장치 상태	0	잠금 상태
	1	개방 상태
도어 모니터링 상태	0	닫힘 상태
	1	개방 상태
	2	모니터링을 하지 않는 상태
단말기 커버 상태	0	닫힘 상태
	1	개방 상태

■ 콜 백 이벤트

UCS 모듈은 단말기로부터 수신된 데이터를 응용프로그램으로 전달 하기 위하여 이벤트 방식을 사용한다.

이벤트	값(상수)
UCSAPI_EVENT	1600
UCSAPI_EVENT_CONNECTED	UCSAPI_EVENT+1
UCSAPI_EVENT_DISCONNECTED	UCSAPI_EVENT+2
UCSAPI_EVENT_VERIFY	UCSAPI_EVENT+3
UCSAPI_EVENT_GETACCESSLOG	UCSAPI_EVENT+4
UCSAPI_EVENT_GETACCESSLOGCOUNT	UCSAPI_EVENT+5
UCSAPI_EVENT_ADDUSER	UCSAPI_EVENT+10
UCSAPI_EVENT_DELETEUSER	UCSAPI_EVENT+11
UCSAPI_EVENT_GETUSERCOUNT	UCSAPI_EVENT+12
UCSAPI_EVENT_GETUSERLIST	UCSAPI_EVENT+13
UCSAPI_EVENT_GETUSERDATA	UCSAPI_EVENT+14
UCSAPI_EVENT_REQUEST_VERIFYINFO	UCSAPI_EVENT+20
UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH	UCSAPI_EVENT+21
UCSAPI_EVENT_REQUEST_VERIFYMATCH	UCSAPI_EVENT+22
UCSAPI_EVENT_REQUEST_IDENTIFYMATCH	UCSAPI_EVENT+23
UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTIO	UCSAPI_EVENT+30

N	
UCSAPI_EVENT_GET_EXPAND_TERMINAL_OPTION	UCSAPI_EVENT+31
UCSAPI_EVENT_SET_TERMINAL_OPTION	UCSAPI_EVENT+32
UCSAPI_EVENT_FW_UPGRADING	UCSAPI_EVENT+80
UCSAPI_EVENT_FW_UPGRADE	UCSAPI_EVENT+81
UCSAPI_EVENT_FW_VERSION	UCSAPI_EVENT+82
UCSAPI_EVENT_TERMINAL_OPEN	UCSAPI_EVENT+90
UCSAPI_EVENT_TERMINAL_STATUS	UCSAPI_EVENT+91

< 콜 백 이벤트 요약 >

단말기 접속 알림 : UCSAPI_EVENT_CONNECTED

UCS 모듈은 단말기가 접속 되었을 때 UCSAPI_EVENT_CONNECTED 이벤트를 응용프로그램으로 통지 한다.

단말기 접속 끊김 통지 : UCSAPI_EVENT_DISCONNECTED

UCS 모듈은 단말기의 접속이 해지 되었을 때 UCSAPI_EVENT_DISCONNECTED 이벤트를 응용프로그램으로 통지 한다.

인증 결과 통지 : UCSAPI_EVENT_VERIFY

UCS 모듈은 단말기가 S/N 모드로 동작하고 사용자 인증을 수행 하였을 때 UCSAPI_EVENT_VERIFY 이벤트를 응용프로그램으로 즉시 통지 한다.

인증 로그에 수신 통지 : UCSAPI_EVENT_GETACCESSLOG

UCS 모듈은 응용프로그램의 인증로그 요청에 대한 응답으로 UCSAPI_EVENT_GETACCESSLOG 이벤트를 응용프로그램으로 통지 한다.

인증 로그 개수 통지 : UCSAPI_EVENT_GETACCESSLOGCOUNT

UCS 모듈은 응용프로그램의 인증로그 개수 요청에 대한 응답으로 UCSAPI_EVENT_GETACCESSLOGCOUNT 이벤트를 응용프로그램으로 통지 한다.

사용자 추가에 대한 결과 통지 : UCSAPI_EVENT_ADDUSER

UCS 모듈은 응용프로그램의 사용자 추가 요청에 대한 응답으로 UCSAPI_EVENT_ADDUSER 이벤트를 응용프로그램으로 통지 한다.

사용자 삭제에 대한 결과 통지 : UCSAPI_EVENT_DELETEUSER

UCS 모듈은 응용프로그램의 사용자 삭제 요청에 대한 응답으로 UCSAPI_EVENT_DELETEUSER 이벤트를 응용프로그램으로 통지 한다.

사용자 개수 통지 : UCSAPI_EVENT_GETUSERCOUNT

UCS 모듈은 응용프로그램의 사용자 개수 요청에 대한 응답으로 UCSAPI_EVENT_GETUSERCOUNT 이벤트를 응용프로그램으로 통지 한다.

사용자 리스트 통지 : UCSAPI_EVENT_GETUSERLIST

UCS 모듈은 응용프로그램의 사용자 리스트 요청에 대한 응답으로 UCSAPI_EVENT_GETUSERLIST 이벤트를 응용프로그램으로 통지 한다.

사용자 데이터 통지 : UCSAPI_EVENT_GETUSERDATA

UCS 모듈은 응용프로그램의 사용자 데이터 요청에 대한 응답으로 UCSAPI_EVENT_GETUSERDATA 이벤트를 응용프로그램으로 통지 한다.

사용자 인증 타입 요청 통지 : UCSAPI_EVENT_REQUEST_VERIFYINFO

UCS 모듈은 단말기의 사용자 인증 타입 요청에 대하여 UCSAPI_EVENT_REQUEST_VERIFYINFO 이벤트를 응용프로그램으로 통지 한다.

카드 사용자 인증 요청 통지 : UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH

UCS 모듈은 단말기의 사용자 카드 사용자 인증 요청에 대하여 UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH 이벤트를 응용프로그램으로 통지 한다.

사용자 1:1 인증 요청 통지 : UCSAPI_EVENT_REQUEST_VERIFYMATCH

UCS 모듈은 단말기의 사용자 1:1 인증 요청에 대하여 UCSAPI_EVENT_REQUEST_VERIFYMATCH 이벤트를 응용프로그램으로 통지 한다.

사용자 1:N 인증 요청 통지 : UCSAPI_EVENT_REQUEST_IDENTIFYMATCH

UCS 모듈은 단말기의 사용자 1:N 인증 요청에 대하여 UCSAPI_EVENT_REQUEST_IDENTIFYMATCH 이벤트를 응용프로그램으로 통지 한다.

단말기 기본 옵션 정보 통지 : UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTION

UCS 모듈은 응용프로그램의 단말기 기본 옵션 정보 요청에 대한 응답으로

UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTION 이벤트를 응용프로그램으로 통지 한다.

단말기 확장 옵션 정보 통지 : UCSAPI_EVENT_GET_EXPAND_TERMINAL_OPTION

UCS 모듈은 응용프로그램의 단말기 확장 옵션 정보 요청에 대한 응답으로 UCSAPI_EVENT_GET_EXPAND_TERMINAL_OPTION 이벤트를 응용프로그램으로 통지 한다.

단말기 옵션 설정 통지 : UCSAPI_EVENT_SET_TERMINAL_OPTION

UCS 모듈은 응용프로그램의 단말기 옵션 설정 요청에 대한 응답으로 UCSAPI_EVENT_SET_TERMINAL_OPTION 이벤트를 응용프로그램으로 통지 한다.

펌웨어 업그레이드 상황 통지 : UCSAPI_EVENT_FW_UPGRADING

UCS 모듈은 응용프로그램의 펌웨어 업그레이드 요청에 대한 응답으로 UCSAPI_EVENT_FW_UPGRADING 이벤트를 응용프로그램으로 통지 한다.

펌웨어 업그레이드 완료 통지 : UCSAPI_EVENT_FW_UPGRADE

UCS 모듈은 응용프로그램의 펌웨어 업그레이드 요청에 대한 응답으로 UCSAPI_EVENT_FW_UPGRADE 이벤트를 응용프로그램으로 통지 한다.

펌웨어 버전 통지 : UCSAPI_EVENT_FW_VERSION

UCS 모듈은 응용프로그램의 펌웨어 버전 요청에 대한 응답으로 UCSAPI_EVENT_FW_VERSION 이벤트를 응용프로그램으로 통지 한다.

단말기 잠금 장치 강제 열림에 대한 결과 통지 : UCSAPI_EVENT_TERMINAL_OPEN

UCS 모듈은 응용프로그램의 사용자 데이터 요청에 대한 응답으로 UCSAPI_EVENT_TERMINAL_OPEN 이벤트를 응용프로그램으로 통지 한다.

단말기 상태 통지 : UCSAPI_EVENT_TERMINAL_STATUS

UCS 모듈은 응용프로그램의 사용자 데이터 요청에 대한 응답으로 UCSAPI_EVENT_TERMINAL_STATUS 이벤트를 응용프로그램으로 통지 한다.

3. 설치

3.1 시스템 요구 사항

■ CPU

Intel Pentium 133Mhz 이상

■ Memory

16M 이상

■ USB Port

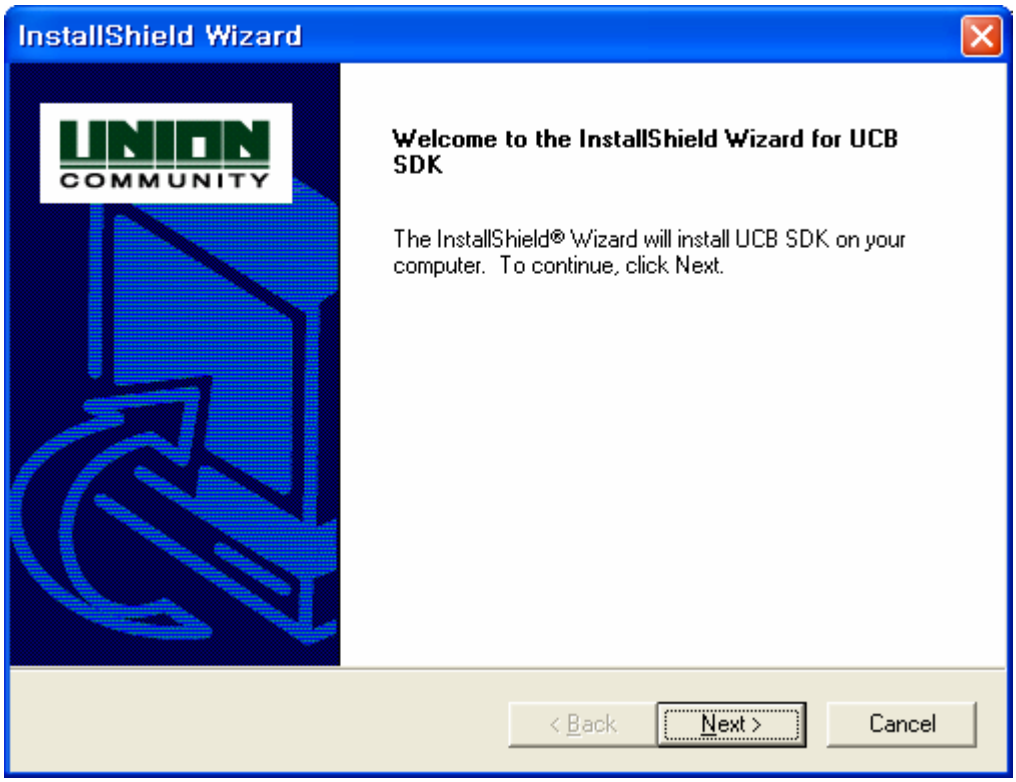
USB 1.1

■ OS

Windows 98/ME 또는 2000/XP/2003/Vista(32 bit only)

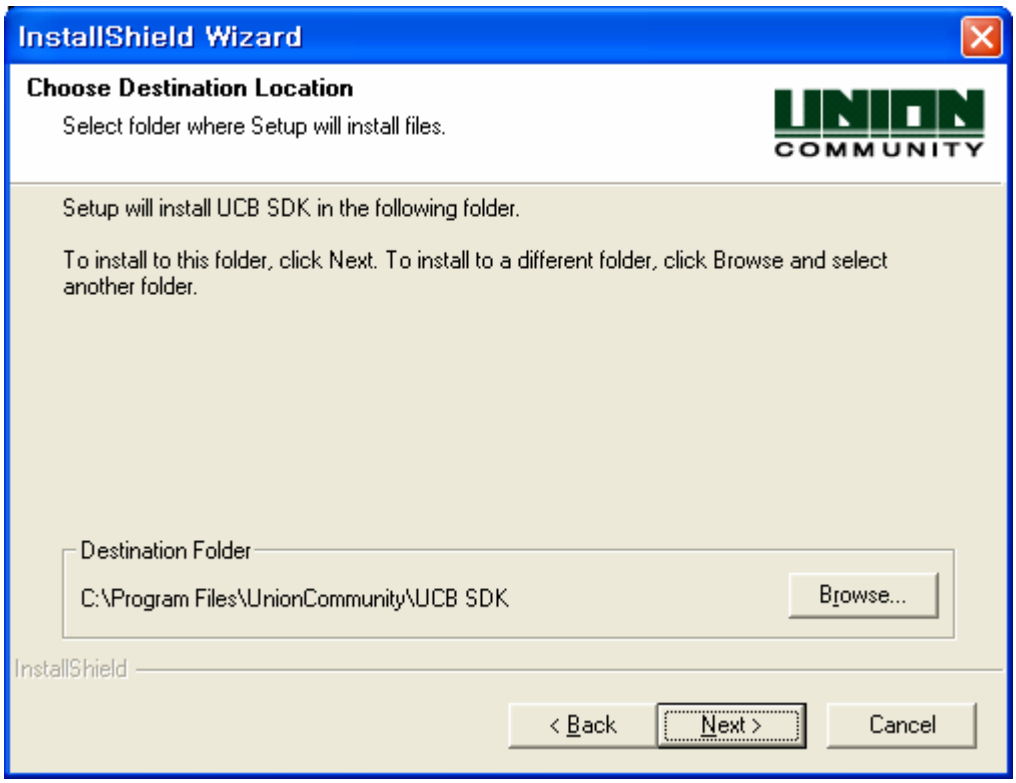
3.2 설치하기

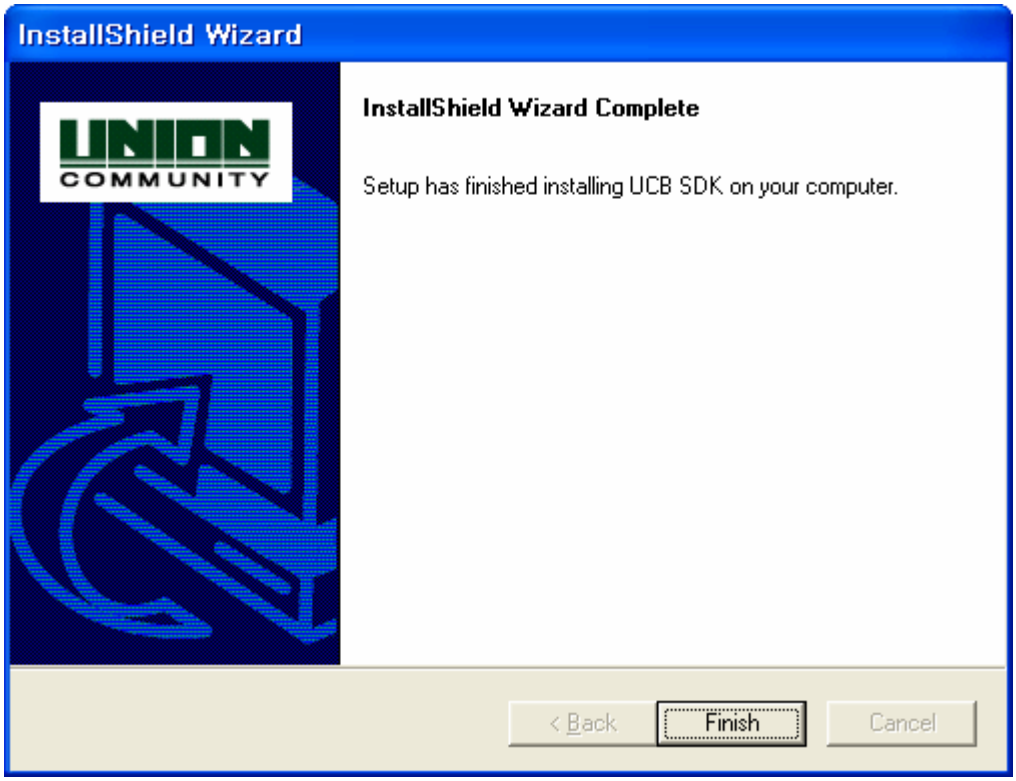
설치 CD를 삽입하면, Setup.exe가 자동으로 실행 됩니다.



단계별로 내용을 확인 하시면서 설치 하시기 바랍니다.

사용자 정보와 제품의 시리얼 번호를 입력 합니다.





3.3 설치되는 파일들

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

3.3.1 Windows System Directory

UCSAPI.dll
네트워크형 지문인식 단말기와 관련된 모든 기능을 구현하고 있는 기본 모듈이다

UCSCOM.dll
RAD Tool 개발자 및 웹 개발자를 위한 COM 모듈

WSEngine.dll
Windows Socket I/O 처리를 위한 통신 모듈

3.3.2 실행모듈 : /Bin

UCSSample.exe
UCSAPI.dll
UCSCOM.dll
WSEngine.dll

3.3.3 헤더 파일 : /Include

ucsapi.h
UCSAPI 모듈의 기본 헤더 파일이다. 어플리케이션은 UCSAPI 모듈을 사용하기 위하여 이 파일을 Include
해야 한다. ucsapi.h는 모듈 사용에 필요한 ucsapi_api.h, ucsapi_err.h, ucsapi_type.h 파일을 추가로 Include 한다.

ucsapi_api.h
UCSAPI 모듈에서 제공하는 모든 함수의 프로토타입이 정의 되어있다.

ucsapi_err.h

UCSAPI 모듈에서 사용하는 에러코드가 정의 되어있다.

ucsapi_type.h

UCSAPI 모듈에서 사용하는 타입과 매크로가 정의 되어있다.

ucsapi_type.h는 ucsapi_porttype.h를 추가로 Include 한다.

ucsapi_porttype.h

UCSAPI 모듈에서 사용하는 타입과 매크로가 정의 되어있다.

3.3.4 라이브러리 파일 : /lib

ucsapi.lib

어플리케이션에서 모듈을 정적으로 Link 하기위한 라이브러리 파일

3.3.5 샘플 소스 : /Sample

Visual C++

UCSAPI.dll을 이용한 샘플 소스

Visual Basic

UCSCOM.dll을 이용한 샘플 소스

Delphi

UCSCOM.dll을 이용한 샘플 소스

3.3.6 매뉴얼 : /Manual

한글/영문 매뉴얼

4. UCSAPI types and macros

UCSAPI

Definition of the UCSAPI calling conventions

```
#ifdef (WIN32)
#define UCSAPI __stdcall
#else
#define UCSAPI
#endif
```

UCSAPI_BOOL

이 타입은 성공 또는 실패의 상태를 표현하기 위해 정의 한다.

```
typedef uint32_t UCSAPI_BOOL;
```

UCSAPI_RETURN

이 타입은 함수의 처리 상태를 반환 하기 위하여 정의 한다.

```
typedef uint32_t UCSAPI_RETURN;

        #define UCSAPI_OK     (0)
```

UCSAPI_EventHandler

이 타입은 단말기로부터 발생하는 이벤트를 받기 위한 콜 백 함수에 대한 정의 이다.

```
typedef UCSAPI_RETURN   (CALLBACK * UCSAPI_EventHandler) (
        uint32_t TerminalID,
```

```
    uint32_t EventType,
    uint32_t wParam,
    uint32_t lParam);
```

UCSAPI_CALLBACK_EVENT

이 타입은 단말기로부터 발생하는 콜 백 이벤트를 정의 한다.

```
typedef uint32_t UCSAPI_CALLBACK_EVENT;

    #define UCSAPI_EVENT                            1600
    #define UCSAPI_EVENT_CONNECTED                  UCSAPI_EVENT+1
    #define UCSAPI_EVENT_DISCONNECTED               UCSAPI_EVENT+2
    #define UCSAPI_EVENT_VERIFY                     UCSAPI_EVENT+3
    #define UCSAPI_EVENT_GETACCESSLOG               UCSAPI_EVENT+4
    #define UCSAPI_EVENT_GETACCESSLOGCOUNT          UCSAPI_EVENT+5
    #define UCSAPI_EVENT_ADDUSER                    UCSAPI_EVENT+10
    #define UCSAPI_EVENT_DELETEUSER                 UCSAPI_EVENT+11
    #define UCSAPI_EVENT_GETUSERCOUNT               UCSAPI_EVENT+12
    #define UCSAPI_EVENT_GETUSERLIST                UCSAPI_EVENT+13
    #define UCSAPI_EVENT_GETUSERDATA                UCSAPI_EVENT+14
    #define UCSAPI_EVENT_REQUEST_VERIFYINFO         UCSAPI_EVENT+20
    #define UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH    UCSAPI_EVENT+21
    #define UCSAPI_EVENT_REQUEST_VERIFYMATCH UCSAPI_EVENT+22
    #define UCSAPI_EVENT_REQUEST_IDENTIFYMATCH      UCSAPI_EVENT+23
    #define UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTION  UCSAPI_EVENT+30
    #define UCSAPI_EVENT_GET_EXPAND_TERMINAL_OPTION UCSAPI_EVENT+31
    #define UCSAPI_EVENT_SET_TERMINAL_OPTION UCSAPI_EVENT+32
    #define UCSAPI_EVENT_FW_UPGRADING               UCSAPI_EVENT+80
    #define UCSAPI_EVENT_FW_UPGRADE                 UCSAPI_EVENT+81
    #define UCSAPI_EVENT_FW_VERSION                 UCSAPI_EVENT+82
    #define UCSAPI_EVENT_TERMINAL_OPEN              UCSAPI_EVENT+90
```

```
#define UCSAPI_EVENT_TERMINAL_STATUS                    UCSAPI_EVENT+91
```

UCSAPI_OPTION_FLAG

이 타입은 단말기 옵션 플래그를 정의 한다.

```
typedef uint32_t UCSAPI_OPTION_FLAG;

#define UCSAPI_OPTION_NAME_BIT                  2
#define UCSAPI_OPTION_NAME              1 << UCSAPI_OPTION_NAME_BIT
#define UCSAPI_OPTION_SERVERINFO_BIT            3
#define UCSAPI_OPTION_SERVERINFO      1 << UCSAPI_OPTION_SERVERINFO_BIT
#define UCSAPI_OPTION_NETWORKINFO_BIT           4
#define UCSAPI_OPTION_NETWORKINFO   1 << UCSAPI_OPTION_NETWORKINFO_BIT
#define UCSAPI_OPTION_WORKINGMODE_BIT           6
#define UCSAPI_OPTION_WORKINGMODE 1 << UCSAPI_OPTION_WORKINGMODE_BIT
#define UCSAPI_OPTION_OPERATIONMODE_BIT     7
#define UCSAPI_OPTION_OPERATIONMODE         1                                   <<
UCSAPI_OPTION_OPERATIONMODE_BIT
```

UCSAPI_LOG_TYPE

이 타입은 로그 오퍼레이션을 위한 타입을 정의 한다.

```
typedef uint32_t UCSAPI_LOG_TYPE;

#define UCSAPI_TYPE_NEW_LOG             0
#define UCSAPI_TYPE_OLD_LOG             1
#define UCSAPI_TYPE_ALL_LOG             2
```

UCSAPI_LENGTH

이 타입은 데이터의 길이를 정의 한다.

typedef uint32_t UCSAPI_LENGTH;

```
#define UCSAPI_LEN_PWD                  8
#define UCSAPI_LEN_CARDNUM              20
```

UCSAPI_AUTH_TYPE

이 타입은 인증 타입을 정의 한다.

typedef uint32_t UCSAPI_AUTHTYPE;

```
#define UCSAPI_AUTHTYPE_FP                                  0
#define UCSAPI_AUTHTYPE_FPCARD                              1
#define UCSAPI_AUTHTYPE_PW                                  2
#define UCSAPI_AUTHTYPE_CARD                                3
#define UCSAPI_AUTHTYPE_CARD_OR_FP                          4
#define UCSAPI_AUTHTYPE_CARD_AND_FP                         5
#define UCSAPI_AUTHTYPE_CARD_OR_PW                          6
#define UCSAPI_AUTHTYPE_CARD_AND_PW                         7
#define UCSAPI_AUTHTYPE_ID_AND_FP_OR_CARD_AND_FP            8
#define UCSAPI_AUTHTYPE_ID_AND_PW_OR_CARD_AND_PW            9
#define UCSAPI_AUTHTYPE_FP_AND_PW                           10
#define UCSAPI_AUTHTYPE_FP_OR_PW                            11
```

UCSAPI_AUTH_REQ

이 타입은 단말기가 서버로 인증 요청 시 지정되는 타입을 정의 한다.

typedef uint32_t UCSAPI_AUTHREQ;

```
#define UCSAPI_AUTHREQ_FP_1TO1                  0
#define UCSAPI_AUTHREQ_FP_1TON                  1
```

```
#define UCSAPI_AUTHREQ_FPCARD          2
#define UCSAPI_AUTHREQ_CARD            3
#define UCSAPI_AUTHREQ_PW              4
```

UCSAPI_AUTH_MODE

이 타입은 인증 모드를 정의 한다.

```
typedef uint32_t UCSAPI_AUTHMODE;

#define UCSAPI_AUTHMODE_ATTENDANCE     0
#define UCSAPI_AUTHMODE_LEAVE          1
#define UCSAPI_AUTHMODE_NORMAL         2
#define UCSAPI_AUTHMODE_OUT            3
#define UCSAPI_AUTHMODE_RETURN         4
```

UCSAPI_PACKET_INFO

이 데이터 타입은 송수신 패킷에 대한 정보를 정의 한다.

```
typedef struct ucsapi_packet_info
{
    uint32_t ClientID;
    uint32_t ErrorCode;
} UCSAPI_PACKET_INFO;
```

ClientID : 작업 요청자 ID. (클라이언트/서버 모델 개발 시에 사용 된다.)
ErrorCode : 작업 처리 결과에 대한 에러 코드.

UCSAPI_DATA

이 데이터 타입은 가변길이의 데이터를 위하여 정의 한다.

```
typedef struct ucsapi_data
```

```
{
        uint32_t   Length;
        void*      Data;
} UCSAPI_DATA;
```

Length : 데이터 길이.
Data : 데이터 버퍼에 대한 포인터.

UCSAPI_PROCESSING_INFO

이 타입은 작업 진행 상황에 대한 정보를 담기 위하여 정의 한다.

```
typedef struct ucsapi_processing_info
{
        int32_t CurrentBlock;
        int32_t   TotalBlock;
} UCSAPI_PROCESSING_INFO;
```

CurrentBlock : 현재 전송 중인 데이터의 인덱스
TotalBlock : 전송 해야 할 총 데이터의 수

UCSAPI_VERIFY_INFO

이 타입은 인증 요청에 대한 정보를 담기 위하여 정의 한다.

```
typedef struct ucsapi_verify_info
{
        uint32_t UserID;
        int32_t   AuthType;          // 0:FP, 1:FP Card, 2:PW, 3:Card
        int32_t   AuthMode;          // 0:출근, 1:퇴근, 2:일반, 3:외근, 4:복귀
        int32_t   InputIdLength;
        int32_t   SecuLevel;
        uint32_t IsAccessibility;
        uint32_t IsMatched;
```

```
    uint32_t ErrorCode;
    UCSAPI_DATA VerifyData;
} UCSAPI_VERIFY_INFO;
```

UserID : 사용자 ID

AuthType : 인증 타입

```
    #define UCSAPI_AUTHTYPE_FP          0
    #define UCSAPI_AUTHTYPE_PW          2
    #define UCSAPI_AUTHTYPE_CARD        3
```

AuthMode : 인증 모드

```
    #define UCSAPI_AUTHMODE_ATTENDANCE      0
    #define UCSAPI_AUTHMODE_LEAVE           1
    #define UCSAPI_AUTHMODE_NORMAL          2
    #define UCSAPI_AUTHMODE_OUT             3
    #define UCSAPI_AUTHMODE_RETURN          4
```

InputIdLength : 단말기에서 서버로 인증 요청 시 Key pad를 통해 눌려진 ID의 길이를 담고있다.
이 값은 1:N 서버 인증 시에 단축 아이디 기능을 구현 하고자 할 때 사용 할 수 있다.
만일 서버에 등록된 사용자 ID가 “1234”이고 단말기로부터 입력된ID(UserID)가 “12”이며 입력된ID 길
이(InputIdLength)가 “2” 인 경우 서버는 1200 부 터 1299까지 99명에 대해서만 비표하면 된다.

SecuLevel : 지문 인증 레벨 (용어정의를 참조 하기 바란다.)

IsAccessibility : 단말기에 인증 권한 여부를 담고 있다. 그 값이 1이면 권한이 있음이고, 0이면 권한이 없는 것이다.

IsMatched :인증 수행 결과를 담고 있다. 그 값이 1이면 권한이 있음이고, 0이면 권한이 없는 것이다.

ErrorCode : 작업 처리에 대한 에러코드를 담고 있다.
UCSAPIERR_NONE 값은 성공을 가리키며 그 외의 모든 다른 값들은 에러 조건을 표현한다.

VerifyData : 인증 요청 시 단말기로부터 획득된 샘플 데이터이다. AuthType에 따라 지문샘플이나 비밀번호, 카드번호가 될 수 있다.

UCSAPI_LOCK_SCHEDULE

이 타입은 단말기의 잠금 기능에 대한 스케줄링 정보를 담고 있다.
단말기는 스케줄링된 일시에 잠금 상태로 진입하거나 잠금 상태로부터 빠져 나올 수 있다. 단말의 잠금 상태에서는 단말의 로그온과 네트웍의 연결은 유지 하지만 서버로부터 가 아닌 모든 지문인식

단말에 대한 접근을 불가능하게 한다.

```
typedef struct ucsapi_lock_schedule
{
        uint16_t Year1;   // Start Year
        uint8_t Month1; // Start Month
        uint8_t Day1;     // Start Day
        uint16_t Year2;   // End Year
        uint8_t Month2; // End Month
        uint8_t Day2;     // End Day
        uint8_t Hour1;    // Start Hour
        uint8_t Min1;     // Start Min
        uint8_t Hour2;    // End Hour
        uint8_t Min2;     // End Min

} UCSAPI_LOCK_SCHEDULE;
```

Year1/Month1/Day1/Hour1/Min1 : 잠금 상태로 진입하기 위한 기간의 시작 정보를 담고 있다.
Year2/Month2/Day2/Hour2/Min2 : 잠금 상태에서 빠져 나오기 위한 기간의 종료 정보를 담고 있다.
만일 기간 정보가 0 으로 채워진다면 시간 정보는 하루 중 잠금 상태의 시작 과 종료 정보를 표현 할 것이다.

Note.
기간 규제가 없는 경우에는 그 값들을 0 으로 채운다. (YYYYMMDDYYYYMMDD)
기간 규제가 없는 상태에서 시간 규제만을 하는 경우에는 잠금 기능은 매일 지정된 시간에 제어 되어 진다.

UCSAPI_OPEN_SCHEDULE

이 타입은 단말기 도어락의 개방 기능에 대한 스케줄링 정보를 담고 있다.
단말기는 스케줄링된 일시에 도어락의 상태를 개방 상태로 진입하거나 개방 상태로부터 빠져 나올

수 있다. 도어락이 개방 상태에서는 단말기의 인증 여부와 관계 없이 출입이 가능하다. 이것은 단말기가 도어락에 대한 통제를 하지 않는 상태를 말한다.

```
typedef struct ucsapi_open_schedule
{
        uint16_t Year1;
        uint8_t Month1;
        uint8_t Day1;
        uint16_t Year2;
        uint8_t Month2;
        uint8_t Day2;
        uint8_t Hour1;
        uint8_t Min1;
        uint8_t Hour2;
        uint8_t Min2;

} UCSAPI_OPEN_SCHEDULE;
```

Year1/Month1/Day1/Hour1/Min1 : 개방 상태로 진입하기 위한 기간의 시작 정보를 담고 있다.
Year2/Month2/Day2/Hour2/Min2 : 개방 상태에서 빠져 나오기 위한 기간의 종료 정보를 담고 있다.
만일 기간 정보가 0 으로 채워진다면 시간 정보는 하루 중 개방 상태의 시작 과 종료 정보를 표현할 것이다.

Note.
기간 규제가 없는 경우에는 그 값들을 0 으로 채운다. (YYYYMMDDYYYYMMDD)
기간 규제가 없는 상태에서 시간 규제만을 하는 경우에는 개방 기능은 매일 지정된 시간에 제어 되어 진다.

UCSAPI_TIMEZONE

이 타입은 타임존에 대한 시간 정보를 담고 있다.

```
typedef struct ucsapi_timezone
{
        uint8_t IsUsed;
        uint8_t StartHour;
        uint8_t StartMin;
        uint8_t EndHour;
        uint8_t EndMin;

} UCSAPI_TIMEZONE;
```

IsUsed : 타임존에 대한 유효성 여부를 담고 있다. 그 값이 1이면 타임존은 유효하다.
StartHour : 시작 시간.
StartMin : 시작 분.
EndHour : 종료 시간.
EndMin : 종료 분.

UCSAPI_DAY_SCHEDULE

이 타입은 요일별 단말기의 잠금 스케줄링 과 도어락의 개방 스케줄에 대한 정보를 담고 있다.
요일별 스케줄링은 하루에 최대 세 개씩 의 잠금, 개방 스케줄링이 가능하다.

```
typedef struct ucsapi_day_schedule
{
        UCSAPI_TIMEZONE Lock1;
        UCSAPI_TIMEZONE Lock2;
        UCSAPI_TIMEZONE Lock3;
        UCSAPI_TIMEZONE Open1;
        UCSAPI_TIMEZONE Open2;
        UCSAPI_TIMEZONE Open3;

} UCSAPI_DAY_SCHEDULE;
```

UCSAPI_DAY_SCHEDULE

이 타입은 휴일에 대한 정보를 담고 있다. 휴일의 종류는 3가지로 표현 된다.

```
typedef struct ucsapi_holiday_schedule
{
        int8_t Month;
        int8_t Day;
        int8_t Type;

} UCSAPI_HOLIDAY_SCHEDULE;
```

Month,Day : 설정된 휴일의 날짜 정보.
Number : 휴일 타입의 인덱스(1,2,3).

UCSAPI_BASIC_SCHEDULE

이 타입은 기본 스케줄에 대한 설정 정보를 담고 있다.

```
typedef struct ucsapi_basic_schedule
{
        UCSAPI_LOCK_SCHEDULE Lock;
        UCSAPI_OPEN_SCHEDULE Open;

} UCSAPI_BASIC_SCHEDULE;
```

UCSAPI_EXPAND_SCHEDULE

이 타입은 확장 스케줄에 대한 설정 정보를 담고 있다.
확장 스케줄은 요일별 잠금 스케줄과 개방 스케줄 및 휴일에 대한 정보를 담고 있다.
휴일은 최대 100개 까지 설정이 가능 하며, 그 유형은 Holiday1, Holiday2, Holiday3에서 설정이 가능하다.

```
typedef struct ucsapi_expand_schedule
{
```

```
        UCSAPI_DAY_SCHEDULE SUN;
        UCSAPI_DAY_SCHEDULE MON;
        UCSAPI_DAY_SCHEDULE TUE;
        UCSAPI_DAY_SCHEDULE WED;
        UCSAPI_DAY_SCHEDULE THU;
        UCSAPI_DAY_SCHEDULE FRI;
        UCSAPI_DAY_SCHEDULE SAT;
        UCSAPI_DAY_SCHEDULE Holiday1;
        UCSAPI_DAY_SCHEDULE Holiday2;
        UCSAPI_DAY_SCHEDULE Holiday3;
        UCSAPI_HOLIDAY_SCHEDULE Holidays[100];

} UCSAPI_EXPAND_SCHEDULE;
```

UCSAPI_SECURITY_LEVEL

이 타입은 지문 인증 레벨에 대한 정보를 담고 있다.
이 값은 1바이트의 크기이며, 상위 4bit는 Verify(1:1) Level이며, 하위 4bit는 Identify(1:N) Level 정보를 담고있다.

```
typedef struct ucsapi_security_level
{
        uint8_t   Verify            :4;
        uint8_t   Identify  :4;
} UCSAPI_SECURITY_LEVEL;
```

UCSAPI_NETWORK_INFO

이 타입은 단말기의 네트워크 정보를 담고 있다.

```
typedef struct ucsapi_network_info
{
        uint8_t NetworkType;
        uint8_t   IP[4];
        uint8_t   Subnet[4];
```

```
        uint8_t   Gateway[4];
} UCSAPI_NETWORK_INFO;
```

NetworkType : IP 유형. 그 값이 0이면 고정 IP 이며, 1이면 유동 IP를 지원한다.
IP : 단말기 IP 정보.
Subnet : Subnet Mask에 대한 정보.
Gateway : Gateway에 대한 정보.

UCSAPI_SERVER_INFO

이 타입은 서버 접속을 위한 네트워크 정보를 담고 있다.

```
typedef struct ucsapi_server_info
{
        uint8_t   IP[4];
        int16_t   Port;
} UCSAPI_SERVER_INFO;
```

IP : 서버 IP.
Port : 서버 접속을 위한 Port 정보.

UCSAPI_TERMINAL_SCHEDULE

단말기의 잠금과 개방 스케줄 정보를 담고 있다.
단말기는 2가지 방법의 잠금과 개방 스케줄링 기능을 제공한다.
기본 스케줄링 기능은 지정된 기간동안 잠금과 개방 제어를 할 수 있으며, 확장 스케줄링 기능은 요일별 잠금과 개방 제어를 할 수 있다.

```
typedef struct ucsapi_terminal_schedule
{
        int32_t ScheduleType; // 0: Basic, 1: Expand
        union
        {
                UCSAPI_BASIC_SCHEDULE Basic;
                UCSAPI_EXPAND_SCHEDULE Expand;
```

```
        }ScheduleData;

} UCSAPI_TERMINAL_SCHEDULE;
```

ScheduleType: ScheduleData의 타입 값이며, 그 값이 0 이면 ScheduleData는 UCSAPI_BASIC_SCHEDULE 값을 저장하고, 1이면 UCSAPI_EXPAND_SCHEDULE 값을 저장 한다.

UCSAPI_BASIC_TERMINAL_OPTIONS

이 타입은 단말기의 기본 옵션 설정 값을 담고 있다.

```
typedef struct ucsapi_basic_terminal_options
{
        int16_t   Bright;            // 0 - 320
        int16_t   Contrast;          // 0 - 360
        int16_t   Gain;              // 1 - 4
        UCSAPI_SECURITY_LEVEL SecuLevel;
        uint8_t   MicLevel;          // 0 - 7
        uint8_t   IdLength;          // 4 - 8
        uint8_t   IsUseAutoEnter;  // 0/1
        uint8_t   IsUseVoice;        // 0/1
        uint8_t Reserved;
        UCSAPI_TERMINAL_SCHEDULE Schedule;
} UCSAPI_BASIC_TERMINAL_OPTIONS;
```

Bright, Contrast, Gatin : 지문인식 센서에 대한 설정 값이다.(이 값은 사용되지 않는다.)
SecuLevel : 지문 인증 레벨 (2.4절의 용어설명을 참조하기 바란다.)
MicLevel : 마이크의 볼륨 레벨(이 값은 사용되지 않는다.)
IDLength : 사용자 ID의 길이.
IsUseAutoEnter : 단말기의 자동 엔터 기능 여부를 표현 하기 위한 값 이다. 이 기능은 IDLength 만큼 Key 입력이 있을 때 Enter Key를 자동으로 입력 시키는 기능이다.
IsUseVoice : 음성의 사용 여부를 표현하기 위한 값이다. 그 값이 1이면 사용하며, 0이면 사용 하지 않는다.
Reserved : 나중을 위하여 예약된 필드이다.
Schedule : 잠금과 개방 스케줄 정보를 담고 있는 구조체.

UCSAPI_EXPAND_TERMINAL_OPTIONS

이 타입은 단말기의 확장 옵션 설정 값을 담고 있다.

```
typedef struct ucsapi_expand_terminal_options
{
        uint8_t   OperationMode;
        uint8_t   WorkingMode;
        uint8_t   Reserved1;
        UCSAPI_NETWORK_INFO NetworkInfo;
        UCSAPI_SERVER_INFO   ServerInfo;
        int8_t    Reserved2[2];
        int8_t    PrintName[32];
} UCSAPI_EXPAND_TERMINAL_OPTIONS;
```

WorkingMode : 단말기의 작업 모드. (2.4절의 용어설명을 참조하기 바란다.)
OperationMode : 단말기의 운영 모드. (2.4절의 용어설명을 참조하기 바란다.)
Reserved1 : 나중을 위하여 예약된 필드이다.
NetworkInfo : 네트워크 정보를 담기 위한 구조체.
ServerInfo : 서버 정보를 담기 위한 구조체.
Reserved2 : 나중을 위하여 예약된 필드이다.
PrintName : 단말기에 연결된 프린터기로 출력할 문자열.

UCSAPI_TERMINAL_OPTIONS

이 타입은 단말기의 옵션 설정 값을 담기 위한 구조체 이다.

```
typedef struct ucsapi_terminal_options
{
        UCSAPI_BASIC_TERMINAL_OPTIONS BasicOptions;
        UCSAPI_EXPAND_TERMINAL_OPTIONS ExpandOptions;
} UCSAPI_TERMINAL_OPTIONS;
```

BasicOptions : 기본 옵션 설정 정보를 담기 위한 구조체.

ExpandOptions : 확장 옵션 설정 정보를 담기 위한 구조체.

UCSAPI_DATETIME_INFO

이 타입은 날짜와 시간 정보를 담기 위한 구조체 이다.

```
typedef struct ucsapi_datetime_info
{
	uint16_t Year;
	uint8_t		Month;
	uint8_t		Day;
	uint8_t		Hour;
	uint8_t		Min;
	uint8_t		Sec;
	uint8_t		Reserved;
} UCSAPI_DATETIME_INFO;
```

UCSAPI_LOG_DATA

이 타입은 인증 기록을 담기 위한 구조체 이다.

```
typedef struct ucsapi_log_data
{
	uint32_t UserID;
	UCSAPI_DATETIME_INFO DateTimeInfo;
	uint8_t		AuthMode;
	uint8_t		AuthType;
	uint8_t		IsMatched;
	uint8_t		Reserved;
	UCSAPI_PROCESSING_INFO Processing;

} UCSAPI_LOG_DATA;
```

UserID : 사용자 ID.
DateTimeInfo : 인증 시각.

AuthMode : 인증 모드.
AuthType : 인증 타입
IsMatched : 인증 결과. 그 값이 1이면 성공을 가리키며, 0이면 실패를 가리킨다.
Reserved : 나중을 위하여 예약된 필드이다.
Processing : 로그 데이터의 전송 상태를 표현하기 위한 구조체 이다.

UCSAPI_USER_FLAG

이 타입은 사용자의 기본정보를 담기 위한 구조체 이다.

```
typedef struct ucsapi_user_flag
{
        uint8_t   Finger        :1;
        uint8_t   FPCard        :1;
        uint8_t   Password      :1;
        uint8_t   Card          :1;
        uint8_t CardID          :1;
        uint8_t Operation       :1;
        uint8_t   Identify  :1;
        uint8_t   Admin         :1;
} UCSAPI_USER_FLAG;
```

구조체의 멤버들에 대한 자세한 설명은 2.4절 용어설명을 참조 하기 바란다.

UCSAPI_ACCESS_DATE

단말기의 접근 가능한 기간에 대한 정보를 담기 위한 구조체 이다.
구조체의 값은 YYYYMMDD(StartYear/StartMon/StartDay)부터 YYYYMMDD(EndYear/EndMon/EndDay)까지의 의미로 사용된다.

```
typedef struct ucsapi_access_date
{
        uint16_t StartYear;
        uint8_t         StartMon;
        uint8_t         StartDay;
```

```
    uint16_t EndYear;
    uint8_t     EndMon;
    uint8_t     EndDay;
} UCSAPI_ACCESS_DATE;
```

UCSAPI_ACCESS_TIME

단말기의 접근 가능한 시간에 대한 정보를 담기 위한 구조체 이다.

```
typedef struct ucsapi_access_time
{
    uint8_t     StartHour;
    uint8_t     StartMin;
    uint8_t     EndHour;
    uint8_t     EndMin;
} UCSAPI_ACCESS_TIME;
```

UCSAPI_USER_INFO

사용자의 인증 권한 정보를 담기 위한 구조체 이다.

```
typedef struct ucsapi_user_info
{
    uint32_t        UserID;
    UCSAPI_USER_FLAG    UserFlag;
    UCSAPI_SECURITY_LEVEL       SecuLevel;
    uint8_t             Reserved[2];
    UCSAPI_DATETIME_INFO DateTimeInfo;
    UCSAPI_ACCESS_DATE  AccessDate;
    UCSAPI_ACCESS_TIME  AccessTime;
    uint8_t             Password[UCSAPI_LEN_PWD];
    uint8_t             CardNumber[UCSAPI_LEN_CARDNUM];

} UCSAPI_USER_INFO;
```

UserID : 사용자 ID (최대 8자리 숫자)
UserFlag : UCSAPI_USER_FLAG 참조.
SecuLevel : 지문 인증 레벨
Reserved : 나중을 위해 예약된 필드
DateTimeInfo : 사용자 등록일
AccessDate : 접근 가능한 날짜
AccessTime : 접근 가능한 시간 (AccessDate가 0으로 채워진 경우
Password : 패스워드 정보 (최대 4자리 숫자로 구성된 문자열)
CardNumber : RFID 정보 (최대 20자리 16진수로 구성된 문자열)

Note.
출입권한 설정을 하지 않을 경우 기간제한정보(AccessDate)와 시간제한정보(AccessTime)은 0으로 채워진다. 기간에 대한 출입 제한이 없는 상태에서 시간에 대한 출입제한만을 하는 경우에는 매일 지정된 시간에 제한 되어 진다.

UCSAPI_USER_DATA

이 타입은 사용자의 인증 권한 정보와 인증 데이터를 담기 위한 구조체 이다.

```
typedef struct ucsapi_user_data
{
        UCSAPI_USER_INFO      UserInfo;
        UCSAPI_DATA           UserData;
} UCSAPI_USER_DATA;
```

UCSAPI_USER_COUNT

이 타입은 단말기에 저장되어 있는 사용의 개수를 담기 위한 구조체이다.
사용자는 관리자 와 일반 사용자로 구분되어 진다. 단말기 관리자는 단말기의 설정 정보를 변경 할 수 있는 권한을 가진다.

```
typedef struct ucsapi_user_count
{
        uint32_t AdminNumber;
```

```
        uint32_t UserNumber;
} UCSAPI_USER_COUNT;
```

AdminNumber : 관리자의 수
UserNumber : 일반 사용자의 수

UCSAPI_TERMINAL_USER

단말기의 사용자 리스트 가져오기 위한 정보를 담기 위한 구조체

```
typedef struct ucsapi_terminal_user
{
        UCSAPI_PROCESSING_INFO Processing;
        uint32_t IsAdmin;
        uint32_t UserID;
} UCSAPI_TERMINAL_USER;
```

Processing : 사용자 리스트의 전송 상태를 표현하기 위한 구조체 이다.
IsAdmin : 단말 관리자 여부. 그 값이 1이면 관리자 이며, 0이면 일반 사용자 이다.
UserID : 사용자 ID.

UCSAPI_IPADDRESS

이 타입은 IP 주소를 담기 위한 구조체 이다.

```
typedef struct ucsapi_ipaddress
{
        uint8_t addr1;
        uint8_t addr2;
        uint8_t addr3;
        uint8_t addr4;
} UCSAPI_IPADDRESS;
```

UCSAPI_TERMINAL_STATUS

이 타입은 단말기의 상태를 담기 위한 구조체 이다.

```
typedef struct ucsapi_terminal_status
{
        int32_t Terminal;
        int32_t  Door;
        int32_t Cover;
} UCSAPI_TERMINAL_STATUS;
```

멤버에 대한 자세한 내용은 2.4절 용어 설명을 참조 하기 바란다.

5. UCSAPI functions

5.1 Initialize Operations

UCSAPI_ServerStart

```
UCSAPI_RETURN UCSAPI UCSAPI_ServerStart(
        IN uint32_t MaxClient,
        IN uint32_t Port,
        IN int32_t   Reserved,
        IN UCSAPI_EventHandler pUCSAPINotifyCallback);
```

설명

이 함수는 UCSAPI 모듈을 초기화 하며, 서버 기능을 실행한다.

파라메터

MaxClient: 최대 연결 단말기 수를 정의한다. 서버는 입력된 클라이언트 수에 따라 내부 메모리 사용량을 조절하며, 클라이언트 연결이 최대 수를 넘을 경우 자동으로 메모리 사용량을 늘려 연결 수를 확보 한다.

Port : 단말기 접속을 위한 통신 포트

Reserved : 나중을 위해 예약된 필드.

pUCSAPINotifyCallback : Event 통지를 위한 콜백 함수 포인터

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

콜벡 이벤트

UCSAPI_EVENT_CONNECTED

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : uint8_t ip[4]; // 단말기 IP

UCSAPI_ServerStop

UCSAPI_RETURN UCSAPI UCSAPI_ServerStop();

설명

이 함수는 연결된 단말기들의 접속을 해지하며, 서버 기능을 종료한다.

파라메터

없음.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

콜벡 이벤트

없음

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : 없음

5.2 Terminal Database Operations

UCSAPI_AddUser

```
UCSAPI_RETURN UCSAPI UCSAPI_AddUser(
        IN int32_t   ClientID,
        IN int32_t   TerminalID,
        IN uint32_t UserID,
        IN int8_t * UserName,
        IN uint32_t IsAdmin,
        IN int8_t * DateLimit,
        IN int8_t * TimeLimit,
        IN uint32_t SecuLevel,
        IN uint32_t AuthType,
        IN int8_t * Password,
        IN int8_t * RFID,
        IN UCBioAPI_EXPORT_DATA_PTR* pFingerData);
```

설명

이 함수는 지정된 단말기로 사용자 정보를 다운로드 한다.
이미 등록 되어 있는 사용자 정보는 덮어 쓰기 한다.

파라메터

ClientID : 클라이언트 ID
TerminalID : 단말기 ID
UserID: 사용자 ID
UserName : 사용자 이름을 담고 있는 버퍼 포인터.
IsAdmin: 단말기 관리자 여부. 그 값이 1이면 단말 관리자이며, 0이면 일반 사용자 이다.
DateLimit: 인증 가능한 기간을 설정 하기 위한 버퍼 포인터. 인증에 대한 기간 제한을 하지 않는 경우는 NULL을 설정한다.
TimeLimit: 인증 가능한 시간을 설정 하기 위한 버퍼 포인터. 인증에 대한 시간 제한을 하지 않는 경우는 NULL을 설정한다. 기간 제한을 하지 않는 경우 매일 지정된 시간에 인증 제한 된다.

SecuLevel : 지문 인증 레벨. 지정 가능 범위는 1부터 9까지 이며, 레벨 값이 낮을 수록 본인 거부율(FRR)은 낮아지고, 타인 인증률(FAR)은 높아진다.
레벨 값이 높을수록 본인 거부율(FRR)은 높아지고, 타인 인증률(FAR)은 낮아진다.
레벨 값이 0인 경우 단말기에서 지정한 레벨 값을 사용 한다.
AuthType: 인증 타입. 인증 방법은 지문, 비밀번호, 카드의 조합이 될 수 있다.

0 : 지문
1 : 지문 카드
2 : 비밀번호
3 : 카드
4 : 카드 or 지문
5 : 카드 and 지문
6 : 카드 or 비밀번호
7 : 카드 and 비밀번호

Password : 비밀번호를 담고있는 버퍼 포인터. AuthType이 비밀번호의 조합인 경우 설정되며, 그렇지 않은 경우는 NULL 값을 설정 한다.
RFID : 카드번호를 담고 있는 버퍼 포인터. AuthType이 카드의 조합인 경우 설정되며. 그렇지 않은 경우는 NULL 값을 설정 한다.
pFingerData : UCBioAPI 모듈로부터 획득된 지문 데이터를 담고 있는 구조체 포인터. AuthType이 지문의 조합인 경우 설정 되며, 그렇지 않은 경우는 NULL 값을 설정 한다.
(자세한 내용은 UCBioBSP SDK를 참조 바란다.)

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE
UCSAPIERR_INVALID_POINTER
UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_ADDUSER

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : 없음

※ 주의사항

UCSAPI_AddUser 함수를 이용하여 여러 사용자를 단말기로 전송 할 경우 반드시 UCSAPI_AddUser 호출후

UCSAPI_EVENT_ADDUSER 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자를 전송 하여야 한다.

UCSAPI_DeleteUser

```
UCSAPI_RETURN UCSAPI UCSAPI_DeleteUser(
        IN int32_t   ClientID,
        IN int32_t   TerminalID,
        IN int32_t   UserID);
```

설명

이 함수는 지정된 단말기로부터 사용자 정보를 삭제 한다.

UserID의 값이 -1인 경우 단말기에 저장된 모든 사용자 정보가 삭제 된다.

파라메터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

UserID : 사용자 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_DELETEUSER

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : 없음

※ 주의사항

UCSAPI_DeleteUser 함수를 이용하여 단말기의 여러 사용자를 삭제 할 경우 반드시 UCSAPI_DeleteUser 호출후 UCSAPI_EVENT_DELETEUSER 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자를 삭제 하여야 한다.

UCSAPI_GetUserCount

```
UCSAPI_RETURN UCSAPI UCSAPI_GetUserCount(
        IN int32_t   ClientID,
        IN int32_t   TerminalID);
```

설명

이 함수는 지정된 단말기로부터 등록된 사용자의 개수를 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GETUSERCOUNT

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_USER_COUNT

UCSAPI_GetUserList

```
UCSAPI_RETURN UCSAPI UCSAPI_GetUserCount(
        IN int32_t   ClientID,
        IN int32_t   TerminalID);
```

설명

이 함수는 지정된 단말기로부터 등록된 모든 사용자의 ID 리스트를 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GETUSERLIST

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_TERMINAL_USER

UCSAPI_GetUserData

```
UCSAPI_RETURN UCSAPI UCSAPI_GetUserData(
        IN int32_t   ClientID,
        IN uint32_t TerminalID
        IN uint32_t UserID);
```

설명

이 함수는 지정된 단말기로부터 지정된 사용자의 데이터를 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

UserID : 사용자 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GETUSERDATA

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_USER_DATA

5.3 Access Log Operations

UCSAPI_GetAccessLog

```
UCSAPI_RETURN UCSAPI UCSAPI_GetAccessLog(
        IN int32_t ClientID,
        IN int32_t TerminalID,
        IN int32_t LogType);
```

설명

이 함수는 지정된 단말기로부터 지정된 타입의 인증로그를 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

LogType : 로그 타입.

0 : 서버로 전송되지 않은 새로운 로그

1 : 이미 서버로 전송된 로그

2 : 단말기에 저장된 모든 로그

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벨 이벤트

UCSAPI_EVENT_GETACCESSLOG

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_LOG_DATA

UCSAPI_GetAccessLogCount

```
UCSAPI_RETURN UCSAPI UCSAPI_GetAccessLogCount(
        IN int32_t ClientID,
        IN int32_t TerminalID,
        IN int32_t LogType);
```

설명

이 함수는 지정된 단말기로부터 지정된 타입의 인증로그 개수를 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

LogType : 로그 타입.

0 : 서버로 전송되지 않은 새로운 로그

1 : 이미 서버로 전송된 로그

2 : 단말기에 저장된 모든 로그

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GETACCESSLOGCOUNT

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : int32_t Count

5.4 Authentication Operations

UCSAPI_ServerResponse

```
UCSAPI_RETURN UCSAPI UCSAPI_ServerResponse(
        IN int32_t TerminalID,
        IN int32_t Message,
        IN int32_t wParam,
        IN int32_t lParam);
```

설명

이 함수는 단말기로부터의 요청 대한 응답을 위하여 호출 된다.
단말기는 N/S(서버 인증 모드)로 동작시 인증 정보를 획득 하기 위하여 응용 프로그램에게 사용자의 인증 정보를 요청 할 수 있다. 또한 단말기는 사용자의 인증을 위하여 입력된 인증 정보(지문, 카드번호, 비밀번호)를 서버로 전송하여 인증 요청을 할 수 있다.

파라메터

TerminalID : 단말기 ID.
Message : 단말기의 요청에 대한 메시지 타입 (자세한 내용은 2.4절 콜벡 이벤트 정리를 참조 바란다.)
  UCSAPI_EVENT_REQUEST_VERIFYINFO
  UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH
  UCSAPI_EVENT_REQUEST_VERIFYMATCH
  UCSAPI_EVENT_REQUEST_IDENTIFYMATCH
wParam : 패킷 정보(UCSAPI_PACKET_INFO)에 대한 구조체 포인터를 위하여 정의 하고 있다.
lParam : 인증 정보(UCSAPI_VERIFY_INFO)에 대한 구조체 포인터를 위하여 정의 하고 있다.

요청 이벤트	설 명
UCSAPI_EVENT_REQUEST_VERIFYINFO (사용자의 인증 정보 요청)	응용 프로그램은 요청된 사용자의 인증 타입(AuthType)과 인증 권한(IsAccessbility)을 단말기로 응답 하여야 한다. 인증 권한이 없는 경우에는 그것에 대한 에러도 함께 응답하여야 한다.
UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH (카드번호 인증 요청)	응용 프로그램은 단말기로부터 획득한 카드번호를 등록 정보와 비교 후 그 결과를 단말기로 응답 하여야 한다.

UCSAPI_EVENT_REQUEST_VERIFYMATCH (1:1 인증 요청)	응용 프로그램은 단말기로부터 획득한 인증 정보(지문,비밀번호)를 등록 정보와 비교 후 그 결과를 단말기로 응답 하여야 한다.
UCSAPI_EVENT_REQUEST_IDENTIFYMATCH (1:N 지문 인증 요청)	응용 프로그램은 단말기로부터 획득한 지문 정보를 등록 지문들과 비교 하여 그 결과를 단말기로 응답 하여야 한다.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜백 이벤트

UCSAPI_EVENT_REQUEST_VERIFYINFO

UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH

UCSAPI_EVENT_REQUEST_VERIFYMATCH

UCSAPI_EVENT_REQUEST_IDENTIFYMATCH

콜백 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_VERIFY_INFO

5.5 Terminal Management Operations

UCSAPI_GetTerminalCount

```
UCSAPI_RETURN UCSAPI UCSAPI_GetTerminalCount(
        IN int32_t* TerminalCount);
```

설명

이 함수는 서버에 접속 되어 있는 단말기의 개수를 얻어 온다.

파라메터

TerminalCount : 단말기의 개수를 반환 받기 위한 버퍼 포인터.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

없음

콜벡 파라미터

없음

UCSAPI_GetFirmwareVersion

```
UCSAPI_RETURN UCSAPI UCSAPI_GetFirmwareVersion(
        IN int32_t ClientID,
        IN int32_t TerminalID);
```

설명

이 함수는 지정된 단말의 펌웨어 버전을 얻어 온다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_FW_VERSION

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_DATA

UCSAPI_UpgradeFirmware

```
UCSAPI_RETURN UCSAPI UCSAPI_UpgradeFirmware(
        IN int32_t ClientID,
        IN int32_t TerminalID,
        IN int8_t * FilePath);
```

설명

이 함수는 지정된 단말의 펌웨어를 업그레이드 한다.

파라메터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

FilePath : 펌웨어 파일의 경로를 담고 있는 버퍼 포인터.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_FW_UPGRADING

UCSAPI_EVENT_FW_UPGRADE

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : UCSAPI_PROCESSING_INFO

UCSAPI_GetBasicTerminalOption

```
UCSAPI_RETURN UCSAPI UCSAPI_GetBasicTerminalOption(
        IN int32_t ClientID,
        IN int32_t TerminalID);
```

설명

이 함수는 단말기로부터 기본 옵션 설정 값을 얻어 온다.
기본 옵션에 대한 내용은 UCSAPI_BASIC_TERMINAL_OPTIONS 구조체에 정의 되어 있다.

파라메터

ClientID : 클라이언트 ID
TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE
UCSAPIERR_INVALID_POINTER
UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTION

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO
lParam : UCSAPI_BASIC_TERMINAL_OPTIONS

UCSAPI_GetExpandTerminalOption

```
UCSAPI_RETURN UCSAPI UCSAPI_GetExpandTerminalOption(
        IN int32_t ClientID,
        IN int32_t TerminalID);
```

설명

이 함수는 단말기로부터 확장 옵션 설정 값을 얻어 온다.
기본 옵션에 대한 내용은 UCSAPI_EXPAND_TERMINAL_OPTIONS 구조체에 정의 되어 있다.

파라메터

ClientID : 클라이언트 ID
TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE
UCSAPIERR_INVALID_POINTER
UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_GET_EVENT_TERMINAL_OPTION

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO
lParam : UCSAPI_EXPAND_TERMINAL_OPTIONS

UCSAPI_SetTerminalOption

```
UCSAPI_RETURN UCSAPI UCSAPI_SetTerminalOption(
        IN int32_t ClientID,
        IN int32_t TerminalID,
        IN int32_t BasicOptionFlag,
        IN int32_t ExpandOptionFlag,
        IN UCSAPI_BASIC_TERMINAL_OPTIONS* BasicOption,
        IN UCSAPI_EXPAND_TERMINAL_OPTIONS* ExpandOption,
        IN UCSAPI_TERMINAL_SCHEDULE* Schedule);
```

설명

이 함수는 단말기의 옵션 값을 변경 한다.

파라메터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID.

BasicOptionFlag : 변경하고자 하는 기본 옵션 항목의 조합된 값이다.

- UCSAPI_OPTION_SCHEDULE
- UCSAPI_OPTION_USEVOICE
- UCSAPI_OPTION_AUTOENTER
- UCSAPI_OPTION_IDLENGTH
- UCSAPI_OPTION_MICLEVEL
- UCSAPI_OPTION_SECURITYLEVEL
- UCSAPI_OPTION_SENSOR

ExpandOptionFlag : 변경하고자 하는 확장 옵션 항목의 조합된 값이다.

- UCSAPI_OPTION_SERVERINFO
- UCSAPI_OPTION_NETWORKINFO
- UCSAPI_OPTION_WORKINGMODE
- UCSAPI_OPTION_OPERATIONMODE
- UCSAPI_OPTION_NAME

BasicOption : UCSAPI_BASIC_TERMINAL_OPTIONS 구조체에 대한 포인터

ExpandOption : UCSAPI_EXPAND_TERMINAL_OPTIONS 구조체에 대한 포인터

Schedule : UCSAPI_TERMINAL_SCHEDULE 구조체에 대한 포인터

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜백 이벤트

UCSAPI_EVENT_SET_TERMINAL_OPTION

콜백 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : 없음

UCSAPI_TerminalOpen

```
UCSAPI_RETURN UCSAPI UCSAPI_OpenTerminal(
        IN int32_t ClientID,
        IN int32_t TerminalID);
```

설명

이 함수는 단말기에 부착된 잠금 장치를 강제 오픈 한다.

파라메터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_INVALID_POINTER

UCSAPIERR_INVALID_TERMINAL

콜벡 이벤트

UCSAPI_EVENT_TERMINAL_OPEN

콜벡 파라미터

wParam : UCSAPI_PACKET_INFO

lParam : 없음

6. UCSCOM method and property

6.1 Properties

각종 Property에 대해 설명한다.

LONG 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.

LONG ConnectionsOfTerminal

Prototype:

```
[ReadOnly]    long          ConnectionsOfTerminal;
```

Description:

서버에 접속된 단말기의 개수를 담고 있다.
이 값은 GetTerminalCount() 메소드 호출 후 얻을 수 있다.

LONG TotalFingerCount

Prototype:

```
[ReadOnly]    long          TotalFingerCount;
```

Description:

등록 지문의 총 손가락 개수를 담고 있다.
반드시 EventGetUserData 호출 후 사용하여야 한다.

Related methods:

EnrollFromTerminal

Related properties:

FingerID

LONG FingerID

Prototype:

[ReadOnly] long FingerID(long nIndex);

Description:

등록 지문의 손가락 ID 정보를 배열로 담고 있다.

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

반드시 EventGetUserData 호출 후 사용하여야 한다.

Related methods:

EnrollFromTerminal

Related properties:

FPSampleDataLength, FPSampleData

LONG SampleNumber

Prototype:

[ReadOnly] long SampleNumber;

Description:

등록 지문의 손가락별 Template의 개수를 담고 있다. 1또는 2의 값이 담기게 된다.

반드시 EventGetUserData 호출 후 사용하여야 한다.

Related methods:

EnrollFromTerminal

Related properties:

FPSampleDataLength, FPSampleData

LONG FPSampleData

Prototype:

[ReadOnly] VARIANT FPSampleData(
long nFingerID,
long SampleNum);

Description:

등록 지문의 손가락별 Template의 Binary stream 데이터를 얻는다.

nFingerID와 SampleNum은 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.

Binary stream 데이터의 길이 값은 FPSampleDataLength Property를 이용해 얻을 수 있다.

반드시 EventGetUserData 호출 후 사용하여야 한다.

Parameters:

nFingerID:

얻어올 손가락 ID 번호.

nSampleNumber:

얻어올 Sample 번호. 0또는 1의 값을 사용한다.

Related methods:

EnrollFromTerminal

Related properties:

FingerID, SampleNumber, FPSampleDataLength

LONG FPSampleDataLength

Prototype:

[ReadOnly] long FPSampleDataLength(
long nFingerID,
long SampleNum);

Description:

등록 지문의 손가락별 Template의 데이터 크기를 얻는다.

nFingerID와 SampleNum은 FingerID 및 SampleNumber Property를 이용해 얻을 수 있

다.
반드시 EventGetUserData 호출 후 사용하여야 한다.

Parameters:

nFingerID:

얻어올 손가락 ID 번호.

nSampleNumber:

얻어올 Sample 번호. 0또는 1의 값을 사용한다.

Related methods:

EnrollFromTerminal

Related properties:

FingerID, SampleNumber, FPSampleData

6.2 Methods

6.2.1 Initialize Operations

ServerStart
(LONG MaxClient,
  LONG Port)

설명

이 메소드는 UCSCOM 모듈을 초기화 하며, 서버 기능을 실행한다.

파라미터

MaxClient: 최대 연결 단말기 수를 정의한다. 서버는 입력된 클라이언트 수에 따라 내부 메모리 사용량을 조절하며, 클라이언트 연결이 최대 수를 넘을 경우 자동으로 메모리 사용량을 늘려 연결 수를 확보 한다.
Port : 단말기 접속을 위한 통신 포트

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventTerminalConnected(LONG TerminalID, BSTR TerminalIP

이벤트 파라미터

TerminalID: 단말기 ID
TerminalIP: 단말기 IP 스트링

ServerStop

설명

이 메소드는 연결된 단말기들의 접속을 해지하며, 서버 기능을 종료한다.

파라미터

없음

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

없음

이벤트 파라미터

없음

6.2.2 Terminal Database Operations

AddMethodPassword(BSTR TextPW)

설명

이 메소드는 사용자 정보에 비밀번호 정보를 추가 하기 위해 사용하며 반드시 AddUser 메소드 전에 호출 되어야 한다.

파라미터

TextPW : 비밀번호를 담고있는 버퍼 포인터. AuthType이 패스워드의 조합인 경우에만 적용 된다.

AddMethodRFID(BSTR TextRFID)

설명

이 메소드는 사용자 정보에 RFID 정보를 추가 하기 위해 사용하며 반드시 AddUser 메소드 전에 호출 되어야 한다.

파라미터

TextRFID : : 카드번호를 담고 있는 버퍼 포인터. AuthType이 카드의 조합인 경우에만 적용 된다.

```
AddMethodFinger
( BOOL bInitiialize,
  LONG nSrcFPDataType,
  LONG nFPDataSize,
  VARIANT FPData1,
  VARIANT FPData2
)
```

설명

이 메소드는 사용자 정보에 지문 정보를 추가 하기 위해 사용하며 반드시 AddUser 메소드 전에 호출 되어야 한다.

파라미터

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. 하지만 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.

FPData1:
추가하고자 하는 Template 데이터. (Binary stream 데이터)

FPData2:
추가하고자 하는 손가락의 두 번째 Template 데이터. (Binary stream 데이터)
이 값은 옵션으로 지정하지 않아도 된다. 그렇게 될 경우 생성된 FIR은 내부적으로 SamplesPerFinger의 값이 1이 된다.

AddUser
(LONG ClientID,
LONG TerminalID,
LONG UserID,
BSTR UserName,
LONG IsAdmin,
BSTR DateLimit,
BSTR TimeLimit,
LONG SecuLevel,
LONG AuthType)

설명
이 메소드는 지정된 단말기로 사용자 정보를 다운로드 한다.
이미 등록 되어 있는 사용자 정보는 덮어 쓰기 한다.

파라미터
ClientID : 클라이언트 ID
TerminalID : 단말기 ID
UserID: 사용자 ID
UserName : 사용자 이름을 담고 있는 버퍼 포인터.

IsAdmin: 단말기 관리자 여부. 그 값이 1이면 단말 관리자이며, 0이면 일반 사용자 이다.
DateLimit: 인증 가능한 기간을 설정 하기 위한 버퍼 포인터. 인증에 대한 기간 제한을 하지 않는 경우는 NULL을 설정한다.
TimeLimit: 인증 가능한 시간을 설정 하기 위한 버퍼 포인터. 인증에 대한 시간 제한을 하지 않는 경우는 NULL을 설정한다. 기간 제한을 하지 않는 경우 매일 지정된 시간에 인증 제한 된다.
SecuLevel : 지문 인증 레벨. 지정 가능 범위는 1부터 9까지 이며, 레벨 값이 낮을 수록 본인 거부율(FRR)은 낮아지고, 타인 인증률(FAR)은 높아진다.
레벨 값이 높을수록 본인 거부율(FRR)은 높아지고, 타인 인증률(FAR)은 낮아진다.
레벨 값이 0인 경우 단말기에서 지정한 레벨 값을 사용 한다.
AuthType: 인증 타입. 인증 방법은 지문, 비밀번호, 카드의 조합이 될 수 있다.

0 : 지문
1 : 지문 카드
2 : 비밀번호
3 : 카드
4 : 카드 or 지문
5 : 카드 and 지문
6 : 카드 or 비밀번호
7 : 카드 and 비밀번호

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventAddUser(LONG Client, LONG TerminalID);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

※ 주의사항

AddUser 메소드를 이용하여 여러 사용자를 단말기로 전송 할 경우 반드시 AddUser 메소드 호출후 EventAddUser 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자를 전송 하여야 한다.

DeleteUser
(LONG ClientID,
  LONG TerminalID,
  LONG UserID)

설명

이 메소드는 지정된 단말기로부터 사용자 데이터를 삭제 한다.
UserID의 값이 -1인 경우 단말기에 저장된 모든 사용자 정보가 삭제 된다.

파라미터

ClientID : 클라이언트 ID
TerminalID : 단말기 ID
UserID: 사용자 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventDeleteUser(LONG Client, LONG TerminalID);

이벤트 파라미터

ClientID : 클라이언트 ID
TerminalID : 단말기 ID

※ 주의사항

DeleteUser 메소드를 이용하여 단말기의 여러 사용자를 삭제 할 경우 반드시 DeleteUser 메소드 호출후 EventDeleteUser 이벤트를 확인하여 정상 처리 되었는지 확인후 다음 사용자를 삭제 하여야 한다.

GetUserCount
(LONG ClientID,
  LONG TerminalID)

설명

이 메소드는 지정된 단말기로부터 등록된 사용자의 개수를 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventGetUserCount(LONG ClientID, LONG TerminalID, LONG AdminCount, LONG UserCount);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

AdminCount : 단말기 관리자의 수

UserCount : 일반 사용자의 수

GetUserList
(LONG ClientID,
  LONG TerminalID)

설명

이 메소드는 지정된 단말기로부터 등록된 모든 사용자의 ID 리스트를 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventGetUserList(LONG ClientID, LONG TerminalID, LONG UserID, LONG IsAdmin, LONG CurrentBlock, LONG TotalBlock);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

UserID : 사용자 ID

IsAdmin : 단말기 관리자 여부 (그 값이 1이면 단말기 관리자 이며, 0이면 일반 사용자 이다)

CurrentBlock : 현재 전송중인 블록 데이터의 인덱스

TotalBlock : 전송 해야 할 전체 블록 데이터의 수

GetUserData
(LONG ClientID,
  LONG TerminalID,
  LONG UserID)

설명

이 메소드는 지정된 단말기로부터 지정된 사용자의 데이터를 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

UserID : 사용자 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

FPSampleData

TotalFingerCount

FingerID

FPSampleDataLength

SampleNumber

이벤트

EventGetUserData(LONG ClientID, LONG TerminalID, LONG UserID, BSTR UserName, LONG IsAdmin, BSTR DateLimit, BSTR TimeLimit, LONG SecuLevel, LONG AuthType, BSTR TextPW, BSTR TextRFID);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

UserID : 사용자 ID

UserName : 사용자 이름

IsAdmin: 단말기 관리자 여부. 그 값이 1이면 단말 관리자이며, 0이면 일반 사용자 이다.

DateLimit: 인증 가능한 기간 값. 인증에 대한 기간 제한을 하지 않는 경우는 NULL 이다.

TimeLimit: 인증 가능한 시간값. 인증에 대한 시간 제한을 하지 않는 경우는 NULL 이다. 기간 제한을 하지 않는 경우 매일 지정된 시간에 인증 제한 된다.

SecuLevel : 지문 인증 레벨. 지정 가능 범위는 1부터 9까지 이며, 레벨 값이 낮을 수록 본인 거부율(FRR)은 낮아지고, 타인 인증률(FAR)은 높아진다.

AuthType: 인증 타입. 인증 방법은 지문, 비밀번호, 카드의 조합이 될 수 있다.

0 : 지문

1 : 지문 카드

2 : 비밀번호

3 : 카드

4 : 카드 or 지문

5 : 카드 and 지문

6 : 카드 or 비밀번호

7 : 카드 and 비밀번호

TextPW : 비밀번호를 담고있는 버퍼 포인터. AuthType이 비밀번호의 조합인 경우 설정되며, 그렇지 않은 경우는 NULL 값을 설정 한다.

TextRFID : 카드번호를 담고 있는 버퍼 포인터. AuthType이 카드의 조합인 경우 설정되며. 그렇지 않은 경우는 NULL 값을 설정 한다.

6.2.3 Access Log Operations

GetAccessLog
(LONG ClientID,
  LONG TerminalID,
  LONG LogType)

설명

이 메소드는 지정된 단말기로부터 지정된 사용자의 데이터를 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

LogType : 인증 로그 타입

0 : 서버로 전송 되지 않은 새로그

1 : 이미 서버로 전송된 로그

2 : 단말기에 저장된 모든 로그

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventGetAccessLog(LONG ClientID, LONG TerminalID, LONG UserID, BSTR AccessTime, LONG AuthMode, LONG AuthType, LONG IsMatched, LONG CurrentBlock, LONG TotalBlock);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

UserID : 사용자 ID

AccessTime : 인증 시간

AuthMode : 인증 모드

AuthType : 인증 타입

0 : 1:1 지문

1 : 1:N 지문

2 : 비밀번호

3 : 카드

IsMatched : 인증 결과

CurrentBlock : 현재 전송중인 블록의 인덱스

TotalBlock : 전송 해야 할 전체 블록 수

GetAccessLogCount
(LONG ClientID,
  LONG TerminalID,
  LONG LogType)

설명

이 메소드는 지정된 단말기로부터 지정된 타입의 인증로그 개수를 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

LogType : 인증 로그 타입

  0 : 서버로 전송 되지 않은 새로그

  1 : 이미 서버로 전송된 로그

  2 : 단말기에 저장된 모든 로그

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventGetAccessLogCount(LONG ClientID, LONG TerminalID, LONG LogCount);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

LogCount : 로그의 개수

6.2.4 Terminal Operations

GetTerminalCount

설명

이 메소드는 서버에 접속 되어 있는 단말기의 개수를 얻어 온다.

파라미터

없음

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

TerminalCount

이벤트

없음

이벤트 파라미터

없음

GetFirmwareVersion
(LONG ClientID,
  LONG TerminalID)

설명

이 메소드는 지정된 단말의 펌웨어 버전을 얻어 온다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

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

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

Version : 펌웨어 버전

UpgradeFirmware
(LONG ClientID,
  LONG TerminalID,
  BSTR FilePath)

설명

이 메소드는 지정된 단말의 펌웨어를 업그레이드 한다.

파라미터

ClientID : 클라이언트 ID.

TerminalID : 단말기 ID.

FilePath : 펌웨어 파일의 경로를 담고 스트링.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventFirmwareUpgrading(LONG ClientID, LONG TerminalID, LONG CurrentBlock, LONG TotalBlock);

EventFirmwareUpgrade(LONG ClientID, LONG TerminalID);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

CurrentBlock : 현재 전송중인 블록 데이터의 인덱스

TotalBlock : 전송 해야 할 전체 블록 데이터의 수

6.2.5 Authentication Operations

ResponseVerifyInfo
(LONG TerminalID,
  LONG UserID,
  LONG AuthType,
  LONG IsAccessibility,
  LONG ErrorCode)

설명

이 메소드는 사용자 인증 정보 요청에 대한 결과를 응답 하기 위하여 호출 한다.
응용 프로그램은 단말기로부터 지정된 사용자의 인증 정보를 단말기로 전송 한다.
이 메소드는 응용 프로그램이 단말기로부터 요청된 사용자의 인증 타입(AuthType)과 인증 권한(IsAccessbility)을 단말기로 응답 하기 위하여 호출 한다. 인증 권한이 없는 경우에는 그것에 대한 에러도 함께 응답하여야 한다.

파라미터

TerminalID : 단말기 ID.
UserID : 사용자 ID
AuthType : 인증 타입
IsAccessibility : 인증 권한 여부, 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
ErrorCode : 인증 권한에 대한 에러를 담고 있다.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventRequestVerifyInfo(LONG TerminalID, LONG UserID);

이벤트 파라미터

TerminalID : 단말기 ID

UserID : 사용자 ID

ResponseCardVerifyMatch
(LONG TerminalID,
  LONG UserID,
  LONG IsAccessibility,
  LONG IsMatched,
  LONG ErrorCode)

설명

이 메소드는 카드 인증 요청에 대한 결과를 응답 하기 위하여 호출 한다.
응용 프로그램은 단말기로부터 획득한 카드 데이터를 등록 정보와 비교 후 그 결과를 단말기로 응답 한다.

파라미터

TerminalID : 단말기 ID.
UserID : 인증된 사용자 ID
IsAccessibility : 인증 권한 여부, 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
IsMatched : 인증 결과. 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
ErrorCode : 인증 권한 및 결과에 대한 에러를 담고 있다.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventRequestCardVerifyMatch(LONG TerminalID, LONG AuthMode, BSTR TextRFID);

이벤트 파라미터

TerminalID : 단말기 ID

AuthMode : 인증 모드

TextRFID : 텍스트 형태의 RFID 스트링

ResponseVerifyMatch
(LONG TerminalID,
  LONG UserID,
  LONG IsAccessibility,
  LONG IsMatched,
  LONG ErrorCode)

설명

이 메소드는 1:1 인증 요청에 대한 결과를 응답 하기 위하여 호출 한다.
응용 프로그램은 단말기로부터 획득한 지문 또는 비밀번호 데이터를 등록 정보와 비교 후 그 결과를 단말기로 응답 한다.

파라미터

TerminalID : 단말기 ID.
UserID : 인증된 사용자 ID
IsAccessibility : 인증 권한 여부, 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
IsMatched : 인증 결과. 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
ErrorCode : 인증 권한 및 결과에 대한 에러를 담고 있다.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventRequestVerifyMatch(LONG TerminalID, LONG UserID, LONG AuthMode, LONG SecuLevel,

LONG AuthType, BSTR VerifyData);

이벤트 파라미터

TerminalID : 단말기 ID

UserID : 사용자 ID

AuthMode : 인증 모드

UCSAPI_AUTHMODE_ATTENDANCE	0
UCSAPI_AUTHMODE_LEAVE	1
UCSAPI_AUTHMODE_NORMAL	2
UCSAPI_AUTHMODE_OUT	3
UCSAPI_AUTHMODE_RETURN	4

SecuLevel : 지문 인증 레벨 (용어정의를 참조 하기 바란다.)

AuthType : 인증 타입

UCSAPI_AUTHTYPE_FP	0
UCSAPI_AUTHTYPE_PW	2
UCSAPI_AUTHTYPE_CARD	3

VerifyData : 인증 타입에 따른 데이터(지문 또는 비밀번호)

ResponseIdentifyMatch
(LONG TerminalID,
  LONG UserID,
  LONG IsAccessibility,
  LONG IsMatched,
  LONG ErrorCode)

설명

이 메소드는 1:N 인증 요청에 대한 결과를 단말기로 전송 하기 위하여 호출 한다.
응용 프로그램은 단말기로부터 획득한 지문 데이터를 등록 정보들과 비교 후 그 결과를 단말기로 응답 한다.

파라미터

TerminalID : 단말기 ID.
UserID : 인증된 사용자 ID
IsAccessibility : 인증 권한 여부, 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
IsMatched : 인증 결과. 그 값이 1이면 인증 권한이 있으면 0이면 인증 권한이 없다.
ErrorCode : 인증 권한 및 결과에 대한 에러를 담고 있다.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED
UCSAPIERR_NOT_ACTIVE
UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventRequestIdentifyMatch(LONG TerminalID, LONG AuthMode, LONG InputIDLength, LONG

SecuLevel, LONG AuthType, BSTR VerifyData);

이벤트 파라미터

TerminalID : 단말기 ID

AuthMode : 인증 모드

InputIDLength : 단말기에서 서버로 인증 요청시 Key pad를 통해 눌려진 ID의 길이를 담고있다.
이 값은 1:N 서버 인증 시에 단축 아이디 기능을 구현 하고자 할 때 사용 할 수 있다.
만일 서버에 등록된 사용자 ID가 “1234”이고 단말기로부터 입력된ID(UserID)가 “12”이며 입력된ID 길
이(InputIdLength)가 “2” 인 경우 서버는 1200 부터 1299까지 99명에 대해서만 비표하면 된다.

SecuLevel : 지문 인증 레벨 (용어정의를 참조 하기 바란다.)

AuthType : 단말기에 획득한 인증 타입

VerifyData : 인증 요청시 단말기로부터 획득된 샘플 데이터이다. AuthType에 따라 지문샘플이나 비밀번호, 카드번호가 될 수 있다.

6.2.6 Terminal Management Operations

OpenTerminal
(LONG ClientID,
  LONG TerminalID)

설명

이 메소드는 단말기에 부착된 잠금 장치를 강제 오픈 한다.

파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID.

반환 값

UCSAPIERR_NONE 값은 성공을 가리킨다. 모든 다른 값들은 에러 조건을 표현한다.

에러

UCSAPIERR_FUNCTION_FAILED

UCSAPIERR_NOT_ACTIVE

UCSAPIERR_START_SERVER

관련 Properties

ErrorCode

이벤트

EventOpenTerminal(LONG ClientID, LONG TerminalID);

이벤트 파라미터

ClientID : 클라이언트 ID

TerminalID : 단말기 ID

7. 프로그래밍 하기

7.1 UCSAPI 모듈 프로그래밍

7.1.1 서버 시작 및 종료 하기

응용 프로그램은 서버 기능을 시작하기 위하여 UCAPI_ServerStart() 함수를 호출 하여야 한다. UCAPI_ServerStart() 함수는 지정된 접속 포트로 단말기의 접속을 항시 대기 하게 된다. 또한 응용 프로그램은 서버의 기능을 종료 하기 위하여 반드시 UCSAPI_ServerStop 함수를 호출 하여야 한다.

서버 시작 하기

```
typedef UCSAPI_RETURN   (CALLBACK * UCSAPI_EventHandler) (
        uint32_t TerminalID,
        uint32_t EventType,
        uint32_t wParam,
        uint32_t lParam);

LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        switch(EventType)
        {
        case UCSAPI_EVENT_CONNECTED:
             break;
        case UCSAPI_EVENT_DISCONNECTED:
             break;
         .
         .
```

```
        .
       }
}

LONG MaxClient = 1;
LONG Port = 2201;
UCSAPI_RETURN ret = UCSAPI_ServerStart(MaxClient,
                                       Port,
                                       0,
                                       (UCSAPI_EventHandler) UCSAPINotifyCallback);
If(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail
```

서버 종료 하기

응용 프로그램은 서버기능을 종료하기 위하여 UCSAPI_ServerStop() 함수를 호출 한다.

```
UCSAPI_RETURN ret = UCSAPI_ServerStop();
If(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail
```

7.1.2 단말기 사용자 관리 하기

UCSAPI 모듈은 응용프로그램이 단말기의 사용자를 추가 하거나 삭제 할 수 있는 기능을 제공 한다.

사용자 추가 하기

응용 프로그램은 사용자 정보를 단말기로 추가 위하여 UCSAPI_AddUser() 함수를 호출 한다.

```
  uint32_t AuthType = UCSAPI_AUTHTYPE_FP;
  uint32_t SecuLevel = 0; //인증레벨이 0인 경우 단말기는 단말기의 인증 레벨 값을 사용한다. 사용
                              자
                          //의 인증 레벨을 별도 지정하고자 한다면 그 값을 0이 아닌 값으로 하
여야
                          //한다.
  uint32_t UserID = 1; // 사용자 ID는 1 ~ 99999999 까지 지정 가능한다.
  UCSAPI_BOOL IsAdmin = FALSE;
  int8_t UserName[32] = {0,};
  int8_t DateLimit[16] = {0,}; //"YYYYMMDDYYYYMMDD"
                               //기간에 대한 인증 제한을 하기 위한 16바이트 길이의 문자열 이
다.
                               //기간에 대한 인증 제한을 하지 않은 경우 그 값을 0으로 채운다.

  int8_t* TimeLimit[8] = {0,}; //"HHMMHHMM"
                               //시간에 대한 인증 제한을 하기 위한 8바이트 길이의 문자열 이다.
                               //시간에 대한 인증 제한을 하지 않은 경우 그 값을 0으로 채운다.

  UCSAPI_RETURN ret = UCSAPI_AddUser(
      m_ClientID,
      TerminalID,
      UserID,
      (int8_t *)UserName,
      IsAdmin,
      (int8_t *)DateLimit, // 기간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
      (int8_t *)TimeLimit, // 시간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
      SecuLevel,
      AuthType,
      pszPassword,
      pszRFID,
      (UCBioAPI_EXPORT_DATA_PTR)pFingerData);



  if(ret == UCSAPIERR_NONE)
```

```
    //Success
else
    //Fail


// UCSAPI_AddUser에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        switch(EventType)
        {
        case UCSAPI_EVENT_ADDUSER:
            if(pi.ErrorCode == 0)
                // Success
            else
                // Fail
            break;
        }
}
```

사용자 삭제 하기

응용 프로그램은 단말기로부터 사용자를 삭제하기 위하여 UCSAPI_DeleteUser() 함수를 호출 한다. 단말기내의 모든 사용자를 삭제 하고자 한다면 UserID를 -1로 설정 하면 된다.

```
UCSAPI_RETURN ret = UCSAPI_DeleteUser(
    m_ClientID,
    TerminalID,
    UserID);



if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail
```

```
// UCSAPI_DeleteUser에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        switch(EventType)
        {
        case UCSAPI_EVENT_DELETEUSER:
            if(pi.ErrorCode == 0)
                // Success
            else
                // Fail
            break;
        }
}
```

사용자 개수 얻어 오기

응용프로그램은 단말기에 등록되어있는 관리자 개수 와 일반 사용자 개수 정보를 얻기 위하여 UCSAPI_GetUserCount() 함수를 호출 한다.

```
UCSAPI_RETURN ret = UCSAPI_GetUserCount(m_ClientID, TerminalID);

if(ret == UCSAPIERR_NONE)
     //Success
else
     //Fail

// UCSAPI_ GetUserCount 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
```

```
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_USER_COUNT UserCount;
        switch(EventType)
        {
        case UCSAPI_EVENT_GETUSERCOUNT:
            if(pi.ErrorCode == 0)
                // Success
                memcpy(&UserCount,                    (UCSAPI_USER_COUNT*)lParam,
sizeof(UCSAPI_USER_COUNT));
            else
                // Fail
            break;
        }
  }
```

사용자 리스트 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자의 ID 리스트 정보를 얻어 오기 위하여 UCSAPI_GetUserList() 함수를 호출 한다.

```
  ret = UCSAPI_GetUserList(m_ClientID, TerminalID);

  if(ret == UCSAPIERR_NONE)
      //Success
  else
      //Fail

  // UCSAPI_ GetUserList에 대한 콜벡 이벤트
  LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
  {
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_TERMINAL_USER Users;
        switch(EventType)
```

```
        {
        case UCSAPI_EVENT_GETUSERLIST:
            if(pi.ErrorCode == 0)
                // Success
                memcpy(&Users,                                    (UCSAPI_TERMINAL_USER*)lParam,
sizeof(UCSAPI_TERMINAL_USER));
            else
                // Fail
            break;
        }
  }
```

사용자 정보 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자 정보를 얻어 오기 위하여 UCSAPI_GetUserData() 함수를 호출 한다.

```
  UCSAPI_RETURN ret = UCSAPI_GetUserData(m_ClientID, TerminalID, UserID);

  if(ret == UCSAPIERR_NONE)
      //Success
  else
      //Fail

  // UCSAPI_ GetUserData 에 대한 콜벡 이벤트
  LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
  {
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_USER_DATA UserData;
        switch(EventType)
        {
        case UCSAPI_EVENT_GETUSERDATA:
            if(pi.ErrorCode == 0)
                // Success
                memcpy(&UserData, (UCSAPI_USER_DATA*)lParam, sizeof(UCSAPI_USER_DATA));
```

```
            else
                // Fail
            break;
        }
    }
```

7.1.3 단말기 로그 관리

단말기는 동작 모드에 따라 로그 저장 방식에 차이가 있다.
먼저 단말기가 N/S 모드로 동작 시에 인증 로그는 응용프로그램에서 저장 하여야 하며, 응용프로그램이 인증 로그 저장에 실패 할 경우에만 단말기에서 저장 된다. 이 방식은 인증에 대한 대부분의 작업을 응용프로그램에 의존하게 되며, 응용프로그램의 데이터베이스가 손상이 되었을 때 로그 복구가 어렵다는 단점이 있다.
반면에 단말기가 S/N 모드로 동작 시에 인증 로그는 단말기에 저장 하게 되며, 동시에 응용 프로그램으로 전송 한다. 이 방식은 두 가지 저장소에 인증 로그를 저장 할 수 있게 하여 로그 손실에 대한 위험을 줄이게 된다.

로그 데이터 얻어

단말기에 저장된 인증 기록을 얻어 오기 위하여 UCSAPI_GetAccessLog() 함수를 호출 한다.
UCSAPI_GetAccessLog() 함수는 단말기로부터 로그데이터를 획득하기 위하여 세번째 파라미터에 전송 받기 위한 로그의 타입을 입력 하여야 한다. 입력 가능은 로그 타입은 UCSAPI_TYPE_ALL_NEW, UCSAPI_TYPE_ALL_OLD, UCSAPI_TYPE_ALL_ALL 이다.
UCSAPI_TYPE_ALL_NEW은 서버로 전송 되지 않은 로그를 얻기 위한 타입이며, UCSAPI_TYPE_ALL_OLD은 이미 서버로 전송된 로그를 얻기 위한 타입니다. 이 둘을 모두 전송 받기 위한다면 UCSAPI_TYPE_ALL_ALL 타입을 사용 하면 된다.

```
UCSAPI_RETURN ret = UCSAPI_GetAccessLog(m_ClientID, TerminalID, UCSAPI_TYPE_ALL_NEW);

if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail

// UCSAPI_GetAccessLog 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
```

```
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_LOG_DATA LogData;
        switch(EventType)
        {
        case UCSAPI_EVENT_GETACCESSLOG:
            if(pi.ErrorCode == 0)
                // Success
                memcpy(&LogData, (UCSAPI_LOG_DATA*)lParam, sizeof(UCSAPI_LOG_DATA));
            else
                // Fail
            break;
        }
  }
```

로그 데이터 개수 얻어 오기

단말기에 저장된 인증 기록의 개수를 얻어 오기 위하여 UCSAPI_GetAccessLogCount() 함수를 호출 한다.

```
  UCSAPI_RETURN ret = UCSAPI_GetAccessLogCount(m_ClientID, TerminalID,
UCSAPI_TYPE_ALL_NEW);

  if(ret == UCSAPIERR_NONE)
      //Success
  else
      //Fail

  // UCSAPI_GetAccessLogCount 에 대한 콜벡 이벤트
  LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
  {
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        uint32_t LogCount;
        switch(EventType)
```

```
        {
        case UCSAPI_EVENT_GETACCESSLOG:
            if(pi.ErrorCode == 0)
                // Success
                LogCount = (uint32_t)lParam;
            else
                // Fail
            break;
        }
}
```

7.1.4 서버 인증 하기

UCSAPI 모듈은 단말기의 요청에 대한 응답을 하기 위한 API를 제공한다.
단말기는 서버로 작업 요청을 하기 위하여 UCSAPI_EVENT_REQUEST 이벤트를 발생하며, 서버는 단말기로부터 발생한 요청 처리 후 그 결과를 단말로 전송 하여야 한다.
이 작업들은 주로 단말기가 서버인증모드(N/S 또는 NO)로 동작하고 있는 동안 발생하며, 응용 프로그램은 서버 인증 기능을 수행 하기 위하여 로컬 데이터베이스에 사용자의 인증 정보를 저장 하고 있어야 한다.
또한 지문 인증을 수행하기 위해서는 지문 등록 및 인증 기능을 제공하는 UCB(UNION COMMUNITY Biometric) SDK를 사용할 필요가 있다.

사용자 인증타입 요청에 응답하기

단말기는 1:1 인증 절차를 수행하기 위하여 사용자의 인증 타입 정보가 필요하며 이를 획득 하기 위하여 서버로 UCSAPI_EVENT_REQUEST_VERIFYINFO 이벤트를 발생한다. 응용프로그램은 요청된 사용자의 이벤트 타입과 인증권한 정보를 을 단말기로 전송 한다. 단말기는 서버로부터 획득한 인증권한 정보가 인증가능(TRUE) 값을 가지면, 사용자의 지문 샘플(또는 비밀번호) 데이터를 채취하여 서버로 인증 요청을 하며, 그렇지 않으면 인증 절차를 인증 실패로 처리하여 종료 한다.

```
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO pi;
        UCSAPI_VERIFY_INFO vi;


        switch(EventType)
        {
        case UCSAPI_EVENT_REQUEST_VERIFYINFO:
            memcpy(&vi, (UCSAPI_VERIFY_INFO*)lParam, sizeof(UCSAPI_VERIFY_INFO));
            PacketInfo.ClientID = 0;
            PacketInfo.ErrorCode = 0;

            vi.AuthType = UCSAPI_AUTHTYPE_FP;
            vi.IsAccessibility = TRUE;
            vi.UserID = SearchedUserID;
```

```
            UCSAPI_ServerResponse(
                TerminalID,

                UCSAPI_EVENT_REQUEST_VERIFYINFO,
                (WPARAM)&PacketInfo,
                (LPARAM)&VerifyInfo);
            break;
        }
}
```

카드 인증 요청에 응답하기

단말기는 카드 인증을 수행하기 위하여 입력된 카드번호와 함께 서버로 UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 카드번보와 저장된 사용자의 카드번호를 비교 하여 그 결과를 단말기로 전송 한다.

```
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO pi;
        UCSAPI_VERIFY_INFO vi;
        CHAR szCardNumer[21] = {0,}
        switch(EventType)
        {
        case UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH:
            memcpy(&vi, (UCSAPI_VERIFY_INFO*)lParam, sizeof(UCSAPI_VERIFY_INFO));
            PacketInfo.ClientID = 0;
            PacketInfo.ErrorCode = 0;
            memcpy(szCardNumber, vi.VerifyData.Data, vi.VerifyData.Length);

            vi.AuthType = UCSAPI_AUTHTYPE_CARD;

            If(IsAccessiility())
            {
                vi.IsAccessibility = TRUE;

                If(strcmp(szCardNumber, ReferenceCardNumber) == 0)
                {
                    vi.IsMatched = TRUE;
                    vi.ErrorCode = 0;
                }
```

```
                else
                {
                    vi.IsMatched = FALSE;
                    vi.ErrorCode = UCSAPIERR_MATCHING;
                }
                vi.UserID = MatchedUserID;
            }
            else
            {
                vi.IsAccessibility = FALSE;
                vi.IsMatched = FALSE;
                vi.ErrorCode = UCSAPIERR_ACCESSIBILITY;
            }

            UCSAPI_ServerResponse(
                    TerminalID,

                    UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH,
                    (WPARAM)&PacketInfo,
                    (LPARAM)&VerifyInfo);
            break;
        }
    }
```

1:1 인증 요청에 응답하기

단말기는 1:1 인증을 수행하기 위하여 입력된 지문 샘플(또는 비밀번호)과 함께 서버로 UCSAPI_EVENT_REQUEST_VERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿을 비교 하여 그 결과를 단말기로 전송 한다.

```
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO pi;
        UCSAPI_VERIFY_INFO vi;
        CHAR szCardNumer[21] = {0,}
        switch(EventType)
        {
        case UCSAPI_EVENT_REQUEST_VERIFYMATCH:
```

```
            memcpy(&vi, (UCSAPI_VERIFY_INFO*)lParam, sizeof(UCSAPI_VERIFY_INFO));
            PacketInfo.ClientID = 0;
            PacketInfo.ErrorCode = 0;
            memcpy(szCardNumber, vi.VerifyData.Data, vi.VerifyData.Length);

            If(IsAccessiility())
            {
                vi.IsAccessibility = TRUE;

                // 인증 수행 시작
                // ret = UCBioAPI_Verify();
                // 인증 수행 완료

                If(ret == UCBioAPIERROR_NONE)
                {
                    vi.IsMatched = TRUE;
                    vi.ErrorCode = 0;
                    vi.UserID = MatchedUserID;
                }
                else
                {
                    vi.IsMatched = FALSE;
                    vi.ErrorCode = UCSAPIERR_MATCHING;
                }
            }
            else
            {
                vi.IsAccessibility = FALSE;
                vi.IsMatched = FALSE;
                vi.ErrorCode = UCSAPIERR_ACCESSIBILITY;
            }

            UCSAPI_ServerResponse(
                    TerminalID,

                    UCSAPI_EVENT_REQUEST_VERIFYMATCH,
                    (WPARAM)&PacketInfo,
                    (LPARAM)&VerifyInfo);
            break;
        }
    }
```

1:N 지문 인증 요청에 응답하기

단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문과 함께 서버로 인증 요청을 하게 되고, 서버는 단말기로부터 획득한 지문 정보와 사용자들의 지문 정보를 비교하여 그 결과를 단말기로 회신

하여야 한다.

단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문 샘플과 함께 서버로 UCSAPI_EVENT_REQUEST_IDENTIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿들을 비교 하여 그 결과를 단말기로 전송 한다.

```
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO pi;
        UCSAPI_VERIFY_INFO vi;
        CHAR szCardNumer[21] = {0,}
        switch(EventType)
        {
        case UCSAPI_EVENT_REQUEST_VERIFYMATCH:
            memcpy(&vi, (UCSAPI_VERIFY_INFO*)lParam, sizeof(UCSAPI_VERIFY_INFO));
            PacketInfo.ClientID = 0;
            PacketInfo.ErrorCode = 0;
            memcpy(szCardNumber, vi.VerifyData.Data, vi.VerifyData.Length);

            If(IsAccessiility())
            {
                vi.IsAccessibility = TRUE;

                // 인증 수행 시작
                // for(int I = 0; I < UserCount; I ++)
                // {
                //     ret = UCBioAPI_Verify();
                //     If(ret == UCBioAPIERROR_NONE)
                //     {
                //         memcpy(&UserID, Payload.Data, Payload.Length);
                //         break;
                //     }
                // 인증 수행 완료

                If(ret == UCBioAPIERROR_NONE)
                {
                    vi.UserID = MatchedUserID;
                    vi.IsMatched = TRUE;
                    vi.ErrorCode = 0;
                }
```

```
                else
                {
                    vi.IsMatched = FALSE;
                    vi.ErrorCode = UCSAPIERR_MATCHING;
                }
            }
            else
            {
                vi.IsAccessibility = FALSE;
                vi.IsMatched = FALSE;
                vi.ErrorCode = UCSAPIERR_ACCESSIBILITY;
            }

            UCSAPI_ServerResponse(
                  TerminalID,

                  UCSAPI_EVENT_REQUEST_VERIFYMATCH,
                  (WPARAM)&PacketInfo,
                  (LPARAM)&VerifyInfo);
            break;
        }
}
```

7.1.5 단말기 관리 하기

서버에 접속중인 단말기 개수 얻기

응용프로그램은 서버에 접속중인 단말기의 개수를 얻기 위하여 UCSAPI_GetTerminalCount() 함수를 호출 한다.

```
uint32_t ConnectionsOfTerminal=0;
UCSAPI_RETURN ret = UCSAPI_GetTerminalCount(&ConnectionsOfTerminal);

if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail
```

단말기 펌웨어 버전 얻기

응용프로그램은 단말기의 펌웨어 버전을 얻기 위하여 UCSAPI_GetFirmareVersion() 함수를 호출 한다.

```
UCSAPI_RETURN ret = UCSAPI_GetFirmwareVersion(m_ClientID, TerminalID);

if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail

// UCSAPI_GetFirmwareVersion 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
```

```
        UCSAPI_DATA* VerData;
        char szVersion[30]={0,};
        switch(EventType)
        {
        case UCSAPI_EVENT_GETACCESSLOG:
            if(pi.ErrorCode == 0)
                // Success
                VerData = (UCSAPI_DATA*)lParam;
                memcpy(szVersion, VerData.Data, VerData.Length);
            else
                // Fail
            break;
        }
}
```

단말기 펌웨어 업그레이드 하기

응용프로그램은 단말기의 펌웨어를 업그레이드 하기 위하여 UCSAPI_UpgradeFirmare() 함수를 호출 한다.

```
char szPathOfFirmware[MAX_PATH+1] = {0,};
// 펌웨어의 경로는 절대 경로이다.
// 여기서 szPathOfFirmware 값을 채운다.
UCSAPI_RETURN ret = UCSAPI_UpgradeFirmware (m_ClientID, TerminalID, szPathOfFirmware);

if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail

// UCSAPI_UpgradeFirmware 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
```

```
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_PROCESSING_INFO Processing;
        switch(EventType)
        {
        case UCSAPI_EVENT_FW_UPGRADING
            if(pi.ErrorCode == 0)
                memcpy(&Processing,                    (UCSAPI_PROCESSING_INFO*)lParam,
sizeof(UCSAPI_PROCESSING_INFO));
            break;
        case UCSAPI_EVENT_FW_UPGRADE:
            if(pi.ErrorCode == 0)
                // Success
            else
                // Fail
            break;
        }
  }
```

단말기 기본 옵션 값 얻어 오기

응용프로그램은 단말기의 기본 옵션 값을 얻어 오기 위하여 UCSAPI_GetBasicTerminalOption() 함수를 호출 한다.

```
  UCSAPI_RETURN ret = UCSAPI_GetBasicTerminalOption(m_ClientID, TerminalID);

  if(ret == UCSAPIERR_NONE)
      //Success
  else
      //Fail

  // UCSAPI_GetBasicTerminalOption 에 대한 콜벡 이벤트
  LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
  {
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_BASIC_TERMINAL_OPTIONS BasicOptions;
```

```
        switch(EventType)
        {
        case UCSAPI_EVENT_GET_BASIC_TERMINAL_OPTION
            if(pi.ErrorCode == 0)
                memcpy(&BasicOptions,(UCSAPI_BASIC_TERMINAL_OPTIONS*)lParam,
                        sizeof(UCSAPI_BASIC_TERMINAL_OPTIONS));
            break;
        }
}
```

단말기 확장 옵션 값 얻어 오기

응용프로그램은 단말기의 확장 옵션 값을 얻어 오기 위하여 UCSAPI_GetExpandTerminalOption() 함수를 호출 한다.

```
UCSAPI_RETURN ret = UCSAPI_GetExpandTerminalOption(m_ClientID, TerminalID);


if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail


// UCSAPI_GetExpandTerminalOption 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_EXPAND_TERMINAL_OPTIONS ExpandOptions;
        switch(EventType)
        {
        case UCSAPI_EVENT_GET_EXPAND_TERMINAL_OPTION
            if(pi.ErrorCode == 0)
                memcpy(&ExpandOptions,(UCSAPI_EXPAND_TERMINAL_OPTIONS*)lParam,
                        sizeof(UCSAPI_EXPAND_TERMINAL_OPTIONS));
            break;
```

```
        }
    }
```

단말기 옵션 값 설정 하기

응용프로그램은 단말기의 옵션 값을 설정 하기 위하여 UCSAPI_SetTerminalOption() 함수를 호출 한다.

```
uint8_t BasicOptionFlag = 0, ExpandOptionFlag = 0;


//옵션 플레그는 설정 가능한 옵션에 대한 항목의 조합으로 설정한다.
BasicOptionFlag |= UCSAPI_OPTION_IDLENGTH | UCSAPI_OPTION_SCHEDULE;
ExpandOptionFlag |= UCSAPI_OPTION_OPERATIONMODE | UCSAPI_OPTION_NAME;

UCSAPI_BASIC_TERMINAL_OPTIONS BasicOption;
memset(&BasicOption, 0, sizeof UCSAPI_BASIC_TERMINAL_OPTIONS);
UCSAPI_EXPAND_TERMINAL_OPTIONS ExpandOption;
memset(&ExpandOption, 0, sizeof UCSAPI_EXPAND_TERMINAL_OPTIONS);
BasicOption.IdLength = 4;
Schedule.ScheduleType = 1;
//아래의 스케쥴링은 9시30분부터 18시30분까지 단말기를 잠그기 위한 설정 예이다.
Schedule.ScheduleData.Expand.THU.Lock1.IsUsed = 1;
Schedule.ScheduleData.Expand.THU.Lock1.StartHour = 9;
Schedule.ScheduleData.Expand.THU.Lock1.StartMin = 30;
Schedule.ScheduleData.Expand.THU.Lock1.EndHour = 18;
Schedule.ScheduleData.Expand.THU.Lock1.EndMin = 30;
ExpandOption.OperationMode = 1;
sprintf((char*)ExpandOption.PrintName, "%s", "UNION");
ret = UCSAPI_SetTerminalOption(
        m_ClientID,
        TerminalID,
        BasicOptionFlag,
        ExpandOptionFlag,
        &BasicOption,
        &ExpandOption,
        &Schedule);
if(ret == UCSAPIERR_NONE)
    //Success
else
```

```
    //Fail

// UCSAPI_SetTerminalOption 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        UCSAPI_EXPAND_TERMINAL_OPTIONS ExpandOptions;
        switch(EventType)
        {
        case UCSAPI_EVENT_SET_TERMINAL_OPTION
            if(pi.ErrorCode == 0)
                //Success
            break;
        }
}
```

단말기 잠금 장치 강제 개방하기

응용프로그램은 단말기에 부착된 잠금 장치의 강제 개방을 위하여 UCSAPI_OpenTerminal() 함수를 호출 한다.

```
UCSAPI_RETURN ret = UCSAPI_OpenTerminal(m_ClientID, TerminalID);

if(ret == UCSAPIERR_NONE)
    //Success
else
    //Fail

// UCSAPI_OpenTerminal 에 대한 콜벡 이벤트
LRESULT CALLBACK UCSAPINotifyCallback(
        UINT TerminalID,
        UINT EventType,
        WPARAM wParam,
```

```
        LPARAM lParam)
{
        UCSAPI_PACKET_INFO* pi;
        pi = (UCSAPI_PACKET_INFO*)wParam;
        switch(EventType)
        {
        case UCSAPI_EVENT_TERMINAL_OPEN
            if(pi.ErrorCode == 0)
                // Success
            break;
        }
}
```

7.2 UCSCOM 모듈 프로그래밍

UCSCOM 모듈은 RAD Tool(Visual Basic, Delphi 등) 개발자 및 웹 개발자(IIS)를 지원 하기 위하여 개발 되었으며, UCSAPI 모듈에서 지원하는 모든 기능을 지원 하지는 않는다.
UCSCOM 모듈의 Method와 Property에 대해서는 6장 UCBioBSPCOM Methods and Properties에 자세히 설명 되어 있다.

7.2.1 비쥬얼 베이직 프로그래밍

1.COM Object 생성 하기

응용 프로그램은 COM 모듈을 사용하기 위해서는 먼저 모듈에 대한 Object를 생성하여야 한다.
응용 프로그램에서 COM 모듈을 참조하기 위하여 Visual Basic의 프로젝트 메뉴의 참조에서 사용 가능한 참조 항목에 UCBioBSPCOM Type Library를 포함 시키도록 한다.

COM Object 생성

```
Dim UCSCOMObj As UCSCOMLib.UCSCOMObj      'Declaration UCSCOM Object
Set UCSCOMObj = New UCSCOMLib.UCSCOMObj 'Object 생성
```

COM Object 소멸

```
Set UCsCOMObj = nothing
```

2. 서버 시작 및 종료 하기

응용 프로그램은 서버 기능을 시작하기 위하여 ServerStart() 메소드를 호출 하여야 한다. ServerStart() 메소드는 지정된 접속 포트로 단말기의 접속을 항시 대기 하게 된다. 또한 응용프로그램은 서버의 기능을 종료 하기 위하여 반드시 ServerStop() 메소드를 호출 하여야 한다.

서버 시작 하기 초기화

```
Dim MaxClient As Long
Dim Port As Long
MaxClient = 1
Port = 2201

UCSCOMObj.ServerStart(MaxClient, 2201)
If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    ' Success
Else
    ' Failed
End If

//관련 이벤트 1
Private Sub UCSCOMObj_EventTerminaConnected(ByVal TerminalID As Long, ByVal TerminalIP As String)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트 2
Private Sub UCSCOMObj_EventTerminaDisconnected(ByVal TerminalID As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If
```

서버 종료 하기

```
UCSCOMObj.ServerStop()
If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    ' Success
Else
    ' Failed
End If
```

3. 단말기 사용자 관리 하기

UCSCOM 모듈은 응용프로그램이 단말기의 사용자를 추가 하거나 삭제 할 수 있는 기능을 제공 한다.

사용자 추가 하기

응용 프로그램은 사용자 정보를 단말기로 추가 위하여 AddUser() 메소드를 호출 한다.

```
Dim AuthType As Long
Dim IsAdmin As Long
Dim DateLimit As String
Dim TimeLimit As String
Dim SecuLevel As Long

AuthType = UCSAPI_AUTHTYPE_FP
IsAdmin = 0
DateLimit = "0000000000000000" 'YYYYMMDDYYYYMMDD
TimeLimit = "00000000"          'HHmmHHmm
SecuLevel = 0                    '단말기의 인증레벨을 사용 할 경우 0을 지정한다.
                                 '인증레벨이 0인 경우 단말기는 단말기의 인증 레벨 값을 사용한다. '사용자의 인증 레벨을 별도 지정하고자 한다면 그 값을 0이 아닌 값
                                 '으로 하여야 한다.

'인증타입이 UCSAPI_AUTHTYPE_PW의 조합 인 경우 사용 된다.
UCSCOMObj.AddMethodPassword(strPassword);
'인증타입이 UCSAPI_AUTHTYPE_CARD의 조합인 경우 사용 된다.
UCSCOMObj.AddMethodRFID(strRFID);
'인증타입이 UCSAPI_AUTHTYPE_FP의 조합인 경우 사용 된다.
UCSCOMObj.AddMethodFinger(bInitialize, nSrcFPDataType, nFPDataSize, FPData1, FPData2);

UCSCOMObj.AddUser(
    ClientID,
    TerminalID,
    UserID,
    UserName,
    IsAdmin,
    DateLimit, ' 기간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
    TimeLimit, ' 시간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
    SecuLevel,
    AuthType)


If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
```

```
    ‘ Success
Else
    ‘ Failed

//관련 이벤트
Private Sub UCSCOMObj_EventAddUser(ByVal ClientID As Long, ByVal TerminalID As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    ‘Success
Else
    ‘Failed
End If
End Sub
```

사용자 삭제 하기

응용 프로그램은 단말기로부터 사용자를 삭제하기 위하여 DeleteUser() 메소드를 호출 한다.
단말기내의 모든 사용자를 삭제 하고자 한다면 UserID를 -1로 설정 하면 된다.

```
UCSCOMObj.DeleteUser(ClientID, TerminalID, UserID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    ‘Success
Else
    ‘Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventDeleteUser(ByVal ClientID As Long, ByVal TerminalID As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    ‘Success
Else
    ‘Failed
End If

End Sub
```

사용자 개수 얻어 오기

응용프로그램은 단말기에 등록되어있는 관리자 개수 와 일반 사용자 개수 정보를 얻기 위하여 GetUserCount() 메소드를 호출 한다.

```
UCSCOMObj. GetUserCount(ClientID, TerminalID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventGetUserCount(ByVal ClientID As Long, ByVal TerminalID As Long,
                                        ByVal AdminCount As Long, ByVal UserCount As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

End Sub
```

사용자 리스트 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자의 ID 리스트 정보를 얻어 오기 위하여 GetUserList() 메소드를 호출 한다.

```
UCSCOMObj. GetUserList(ClientID, TerminalID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If
```

```
//관련 이벤트
Private Sub UCSCOMObj_EventGetUserList(ByVal ClientID As Long, ByVal TerminalID As Long,
                                       ByVal UserID As Long, ByVal Admin As Long, ByVal
                                       CurrentBlock As Long, ByVal TotalBlock As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Fail
End If

End Sub
```

사용자 데이터 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자 정보를 얻어 오기 위하여 GetUserData() 메소드를 호출 한다.

```
UCSCOMObj. GetUserData(ClientID, TerminalID, UserID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventGetUserData(ByVal ClientID As Long, ByVal TerminalID As Long,
                                       ByVal UserID As Long, ByVal UserName As String, ByVal
                                       Admin As Long, ByVal DateLimit As String, ByVal
                                       TimeLimit As String, ByVal SecuLevel As Long, ByVal
                                       AuthType As Long, ByVal TextPW As String, ByVal
                                       TextRFID As String)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If
```

```
End Sub
```

4. 단말기 로그 관리 하기

단말기는 동작 모드에 따라 로그 저장 방식에 차이가 있다.

먼저 단말기가 N/S 모드로 동작 시에 인증 로그는 응용프로그램에서 저장 하여야 하며, 응용프로그램이 인증 로그 저장에 실패 할 경우에만 단말기에서 저장 된다. 이 방식은 인증에 대한 대부분의 작업을 응용프로그램에 의존하게 되며, 응용프로그램의 데이터베이스가 손상이 되었을 때 로그 복구가 어렵다는 단점이 있다.

반면에 단말기가 S/N 모드로 동작 시에 인증 로그는 단말기에 저장 하게 되며, 동시에 응용 프로그램으로 전송 한다. 이 방식은 두 가지 저장소에 인증 로그를 저장 할 수 있게 하여 로그 손실에 대한 위험을 줄이게 된다.

로그 데이터 얻어 오기

단말기에 저장된 인증 기록을 얻어 오기 위하여 GetAccessLog() 메소드를 호출 한다.

GetAccessLog() 메소드는 단말기로부터 로그데이터를 획득하기 위하여 세번째 파라미터에 전송 받기 위한 로그의 타입을 입력 하여야 한다. 입력 가능은 로그 타입은 UCSAPI_TYPE_ALL_NEW, UCSAPI_TYPE_ALL_OLD, UCSAPI_TYPE_ALL_ALL 이다.

UCSAPI_TYPE_ALL_NEW은 서버로 전송 되지 않은 로그를 얻기 위한 타입이며, UCSAPI_TYPE_ALL_OLD은 이미 서버로 전송된 로그를 얻기 위한 타입니다. 이 둘을 모두 전송 받기 위한다면 UCSAPI_TYPE_ALL_ALL 타입을 사용 하면 된다.

```
UCSCOMObj. GetAccessLog(ClientID, TerminalID, UCSAPI_TYPE_ALL_NEW)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventGetAccessLog(ByVal Handle As Long, ByVal TerminalID As Long,
                                        ByVal UserID As Long, ByVal AccessTime As String, ByVal
                                        AuthMode As Long, ByVal AuthType As Long, ByVal Result
                                        As Long, ByVal CurrentBlock As Long, ByVal TotalBlock As
                                        Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
```

```
Else
    'Failed
End If

End Sub
```

로그 데이터 개수 얻어 오기

단말기에 저장된 인증 기록의 개수를 얻어 오기 위하여 GetAccessLogCount() 메소드를 호출 한다.

```
UCSCOMObj. GetAccessLogCount(ClientID, TerminalID, UCSAPI_TYPE_ALL_NEW)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventGetAccessLogCount(ByVal Handle As Long, ByVal TerminalID As
                                        Long, ByVal LogCount As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

End Sub
```

5. 서버 인증 하기

단말기는 서버로 작업 요청을 하고 서버는 요청된 작업을 처리 후 단말로 그 결과를 전송 하여야 한다.

이 작업들은 주로 단말기가 서버인증모드(N/S 또는 NO)로 동작하고 있는 동안 발생하며, 서버는 인증 수행을 위하여 로컬 데이터베이스에 인증을 위한 사용자 정보를 저장 하고 있어야 한다.

또한 지문 인증을 수행하기 위해서는 지문 등록 및 인증 기능을 제공하는 UCB(UNION COMMUNITY Biometric) SDK를 사용할 필요가 있다.

사용자 인증타입 요청에 응답하기

단말기는 1:1 인증 절차를 수행하기 위하여 사용자의 인증 타입 정보가 필요하며 이를 획득 하기 위하여 서버로 UCSAPI_EVENT_REQUEST_VERIFYINFO 이벤트를 발생한다. 응용프로그램은 요청된 사용자의 이벤트 타입과 인증권한 정보를 을 단말기로 전송 한다. 단말기는 서버로부터 획득한 인증권한 정보가 인증가능(TRUE) 값을 가지면, 사용자의 지문 샘플(또는 비밀번호) 데이터를 채취하여 서버로 인증 요청을 하며, 그렇지 않으면 인증 절차를 인증 실패로 처리하여 종료 한다.

```
  Private Sub UCSCOMObj_EventRequestVerifyInfo(ByVal TerminalID As Long, ByVal UserID As Long)

  Dim AuthType As Long
  Dim IsAccessibility As Long
  Dim ErrorCode As Long
  Dim UserID As Long

  AuthType = UCSAPI_AUTHTYPE_FP ' 사용자의 인증타입이 지문인경우.
  IsAccessibility = 1 '인증 권한 있음. 인증권한이 없는 경우 0 이다.
  ErrorCode = 0 ' 에러 없음
  ResponseVerifyInfo(TerminalID, UserID, AuthType, IsAccessibility, ErrorCode)


  If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
      'Success
  Else
      'Failed
  End If

  End Sub
```

카드 인증 요청에 응답하기

단말기는 카드 인증을 수행하기 위하여 입력된 카드번호와 함께 서버로 UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 카드번보와 저장된 사용자의 카드번호를 비교 하여 그 결과를 단말기로 전송 한다.

```
Private Sub UCSCOMObj_EventRequestCardVerifyMatch(ByVal TerminalID As Long, ByVal AuthMode As Long, ByVal TextRFID As String)

Dim AuthType As Long
Dim IsAccessibility As Long
Dim IsMatched As Long
Dim ErrorCode As Long
Dim MatchedUserID As Long

AuthType = UCSAPI_AUTHTYPE_FP ' 사용자의 인증타입이 지문인경우.
IsAccessibility = 1 '인증 권한 있음. 인증권한이 없는 경우 0 이다.
IsMatched = 1 '인증 수행 결과가 성공인 경우
ErrorCode = 0 ' 에러 없음
ResponseCardVerifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched, ErrorCode)


If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

End Sub
```

1:1 인증 요청에 응답하기

단말기는 1:1 인증을 수행하기 위하여 입력된 지문 샘플(또는 비밀번호)과 함께 서버로 UCSAPI_EVENT_REQUEST_VERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿을 비교 하여 그 결과를 단말기로 전송 한다.

```
Private Sub UCSCOMObj_EventRequestVerifyMatch(ByVal TerminalID As Long, ByVal UserID As Long, ByVal AuthMode As Long, ByVal SecuLevel As Long, ByVal AuthType As Long, ByVal VerifyData As String)

Dim AuthType As Long
Dim IsAccessibility As Long
Dim IsMatched As Long
Dim ErrorCode As Long
Dim MatchedUserID As Long
```

```
  AuthType = UCSAPI_AUTHTYPE_FP ' 사용자의 인증타입이 지문인경우.
  IsAccessibility = 1 '인증 권한 있음. 인증권한이 없는 경우 0 이다.
  IsMatched = 1 '인증 수행 결과가 성공인 경우
  ErrorCode = 0 ' 에러 없음
  ResponseVerifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched, ErrorCode)

  If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
      'Success
  Else
      'Failed
  End If

End Sub
```

1:N 지문 인증 요청에 응답하기

단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문과 함께 서버로 인증 요청을 하게 되고, 서버는 단말기로부터 획득한 지문 정보와 사용자들의 지문 정보를 비교하여 그 결과를 단말기로 회신하여야 한다.

단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문 샘플과 함께 서버로 UCSAPI_EVENT_REQUEST_IDENTIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿들을 비교 하여 그 결과를 단말기로 전송 한다.

```
  Private Sub UCSCOMObj_EventRequestIdentifyMatch(ByVal TerminalID As Long, ByVal AuthMode
  As Long, ByVal InputUserIDLength As Long, ByVal SecuLevel As Long, ByVal AuthType As Long,
  ByVal VerifyData As String)

  Dim AuthType As Long
  Dim IsAccessibility As Long
  Dim IsMatched As Long
  Dim ErrorCode As Long
  Dim MatchedUserID As Long

  AuthType = UCSAPI_AUTHTYPE_FP ' 사용자의 인증타입이 지문인경우.
  IsAccessibility = 1 '인증 권한 있음. 인증권한이 없는 경우 0 이다.
  IsMatched = 1 '인증 수행 결과가 성공인 경우
  ErrorCode = 0 ' 에러 없음

  'MatchedUserID는 인증 성공된 사용자 ID 이다.
  ResponseIdentifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched, ErrorCode)

  If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
```

```
        'Success
    Else
        'Failed
    End If

End Sub
```

6. 단말기 관리 하기

서버에 접속중인 단말기 개수 얻기

응용프로그램은 서버에 접속중인 단말기의 개수를 얻기 위하여 GetTerminalCount() 메소드를 호출 한다.

```
UCSCOMObj.GetTerminalCount()

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
없음
```

단말기 펌웨어 버전 얻기

응용프로그램은 단말기의 펌웨어 버전을 얻기 위하여 GetFirmareVersion() 메소드를 호출 한다.

```
UCSCOMObj.GetFirmware(ClientID, TerminalID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

//관련 이벤트
Private Sub UCSCOMObj_EventFirmwareVersion(ByVal ClientID As Long, ByVal TerminalID As
                                    Long, ByVal Version As String)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
```

```
Else
    'Failed
End If

End Sub
```

단말기 펌웨어 업그레이드 하기

응용프로그램은 단말기의 펌웨어를 업그레이드 하기 위하여 UpgradeFirmare() 함수를 호출 한다.

```
UCSCOMObj.GetFirmware(ClientID, TerminalID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

'관련 이벤트 1 (업그레이드 진행중 이벤트)
Private Sub UCSCOMObj_EventFirmwareUpgrading(ByVal ClientID As Long, ByVal TerminalID As
                                    Long, ByVal CurrentBlock As Long, ByVal TotalBlock As
                                    Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

End Sub

'관련 이벤트 2 (업그레이드 완료 이벤트)
Private Sub UCSCOMObj_EventFirmwareUpgrade(ByVal ClientID As Long, ByVal TerminalID As
                                    Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
```

```
End If
End Sub
```

단말기 옵션 값 얻기 및 설정 하기

UCSCOM 모듈은 단말기 옵션을 관리하기 위한 기능 들은 지원 하지 않는다.

단말기 잠금장치 강제 개방하기

응용프로그램은 단말기에 부착된 잠금 장치의 강제 개방을 위하여 OpenTerminal() 메소드를 호출한다.

```
UCSCOMObj.OpenTerminal(ClientID, TerminalID)

If UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

'관련 이벤트
Private Sub UCSCOMObj_EventOpenTerminal(ByVal ClientID As Long, ByVal TerminalID As Long)
If UCSCOMObj.EventError = UCSAPIERR_NONE Then
    'Success
Else
    'Failed
End If

End Sub
```

7.2.2 델파이 프로그래밍

1. COM Object 생성 하기

COM 모듈을 사용하기 위해서는 먼저 모듈에 대한 Object를 생성하여 한다.

COM Object 생성

```
UCSCOMObj : Variant;
UCSCOMObj := CreateOleObject('UCSCOM.UCSCOMObj');
```

COM Object 소멸

```
//사용이 끝난 Object는 삭제 한다.
UCSCOMObj := null;
Set UCSCOMObj = nothing;
```

2. 서버 시작 및 종료 하기

응용 프로그램은 서버 기능을 시작하기 위하여 ServerStart() 메소드를 호출 하여야 한다. ServerStart() 메소드는 지정된 접속 포트로 단말기의 접속을 항시 대기 하게 된다. 또한 응용프로그램은 서버의 기능을 종료 하기 위하여 반드시 ServerStop() 메소드를 호출 하여야 한다.

서버 시작 하기 초기화

```
Var MaxClient: Integer;
Var Port: Integer;

MaxClient = 1;
Port = 2201;

UCSCOMObj.ServerStart(MaxClient, 2201);
if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;

// 관련 이벤트
procedure EventTerminalConnected(TerminalID: Integer, TerminalIP: WideString);
procedure EventTerminalDisconnected(TerminalID: Integer);
```

서버 종료 하기

```
UCSCOMObj.ServerStop();

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;
```

3. 단말기 사용자 관리 하기

UCSCOM 모듈은 응용프로그램이 단말기의 사용자를 추가 하거나 삭제 할 수 있는 기능을 제공 한다.

사용자 추가 하기

응용 프로그램은 사용자 정보를 단말기로 추가 위하여 AddUser() 메소드를 호출 한다.

```
Var AuthType : Long;
Var IsAdmin : Long;
Var DateLimit : String;
Var TimeLimit : String;
Var SecuLevel : Long;

AuthType = UCSAPI_AUTHTYPE_FP;
IsAdmin = 0;
DateLimit = "0000000000000000"; //YYYYMMDDYYYYMMDD
TimeLimit = "00000000";          //HHmmHHmm
SecuLevel = 0;                   //단말기의 인증레벨을 사용 할 경우 0을 지정한다.
                                 //인증레벨이 0인 경우 단말기는 단말기의 인증 레벨 값을 사용
                                 한다. //사용자의 인증 레벨을 별도 지정하고자 한다면 그 값을
                                 0이 아닌 값
                                 //으로 하여야 한다.

//인증타입이 UCSAPI_AUTHTYPE_PW의 조합 인 경우 사용 된다.
UCSCOMObj.AddMethodPassword(strPassword);
//인증타입이 UCSAPI_AUTHTYPE_CARD의 조합인 경우 사용 된다.
UCSCOMObj.AddMethodRFID(strRFID);
//인증타입이 UCSAPI_AUTHTYPE_FP의 조합인 경우 사용 된다.
UCSCOMObj.AddMethodFinger(bInitialize, nSrcFPDataType, nFPDataSize, FPData1, FPData2);

UCSCOMObj.AddUser(
    ClientID,
    TerminalID,
    UserID,
    UserName,
    IsAdmin,
    DateLimit, // 기간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
    TimeLimit, // 시간에 대한 인증 제한을 하지 않은 경우 NULL을 입력할 수 있다.
    SecuLevel,
    AuthType);
```

```
if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;

//관련 이벤트
procedure EventAddUser(ClientID: Integer, TerminalID: Integer);
```

사용자 삭제 하기

응용 프로그램은 단말기로부터 사용자를 삭제하기 위하여 DeleteUser() 메소드를 호출 한다.
단말기내의 모든 사용자를 삭제 하고자 한다면 UserID를 -1로 설정 하면 된다.

```
UCSCOMObj.DeleteUser(ClientID, TerminalID, UserID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;


//관련 이벤트
procedure EventDeleteUser(ClientID: Integer, TerminalID: Integer, UserID: Integer);
```

사용자 개수 얻어 오기

응용프로그램은 단말기에 등록되어있는 관리자 개수 와 일반 사용자 개수 정보를 얻기 위하여 GetUserCount() 메소드를 호출 한다.

```
UCSCOMObj.GetUserCount(ClientID, TerminalID)

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;
```

```
//관련 이벤트
procedure EventGetUserCount(ClientID: Integer; TerminalID: Integer; AdminCount: Integer;
                                                                    UserCount: Integer);
```

사용자 리스트 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자의 ID 리스트 정보를 얻어 오기 위하여 GetUserList() 메소드를 호출 한다.

```
UCSCOMObj.GetUserList(ClientID, TerminalID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;


//관련 이벤트
procedure EventGetUserList(ClientID: Integer; TerminalID: Integer; UserID: Integer;
                                Admin: Integer; CurrentBlock: Integer; TotalBlock: Integer);
```

사용자 데이터 얻어 오기

응용프로그램은 단말기에 등록되어있는 사용자 정보를 얻어 오기 위하여 GetUserData() 메소드를 호출 한다.

```
UCSCOMObj.GetUserData(ClientID, TerminalID, UserID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
    //Success
    end;


//관련 이벤트
```

```
procedure EventGetUserData(ClientID: Integer; TerminalID: Integer; UserID: Integer;
                           const UserName: WideString; Admin: Integer;
                           const DateLimit: WideString; const TimeLimit: WideString;
                           SecuLevel: Integer; AuthType: Integer; const TextPW: WideString;
                           const TextRFID: WideString);
```

4. 단말기 로그 관리 하기

단말기는 동작 모드에 따라 로그 저장 방식에 차이가 있다.

먼저 단말기가 N/S 모드로 동작 시에 인증 로그는 응용프로그램에서 저장 하여야 하며, 응용프로그램이 인증 로그 저장에 실패 할 경우에만 단말기에서 저장 된다. 이 방식은 인증에 대한 대부분의 작업을 응용프로그램에 의존하게 되며, 응용프로그램의 데이터베이스가 손상이 되었을 때 로그 복구가 어렵다는 단점이 있다.

반면에 단말기가 S/N 모드로 동작 시에 인증 로그는 단말기에 저장 하게 되며, 동시에 응용 프로그램으로 전송 한다. 이 방식은 두 가지 저장소에 인증 로그를 저장 할 수 있게 하여 로그 손실에 대한 위험을 줄이게 된다.

로그 데이터 얻어 오기

단말기에 저장된 인증 기록을 얻어 오기 위하여 GetAccessLog() 메소드를 호출 한다.

GetAccessLog() 메소드는 단말기로부터 로그데이터를 획득하기 위하여 세번째 파라미터에 전송 받기 위한 로그의 타입을 입력 하여야 한다. 입력 가능은 로그 타입은 UCSAPI_TYPE_ALL_NEW, UCSAPI_TYPE_ALL_OLD, UCSAPI_TYPE_ALL_ALL 이다.

UCSAPI_TYPE_ALL_NEW은 서버로 전송 되지 않은 로그를 얻기 위한 타입이며, UCSAPI_TYPE_ALL_OLD은 이미 서버로 전송된 로그를 얻기 위한 타입니다. 이 둘을 모두 전송 받기 위한다면 UCSAPI_TYPE_ALL_ALL 타입을 사용 하면 된다.

```
UCSCOMObj.GetAccessLog(ClientID, TerminalID, UCSAPI_TYPE_ALL_NEW);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;


//관련 이벤트
procedure EventGetAccessLog(ClientID: Integer; TerminalID: Integer; UserID: Integer;
                            const AccessTime: WideString; AuthMode: Integer; AuthType:
                                Integer;
                            Result: Integer; CurrentBlock: Integer; TotalBlock: Integer);
```

로그 데이터 개수 얻어 오기

단말기에 저장된 인증 기록의 개수를 얻어 오기 위하여 GetAccessLogCount() 메소드를 호출 한다.

```
UCSCOMObj.GetAccessLogCount(ClientID, TerminalID, UCSAPI_TYPE_ALL_NEW)

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;


//관련 이벤트
procedure EventGetAccessLogCount(ClientID: Integer; TerminalID: Integer; LogCount: Integer);
```

5. 서버 인증 하기

단말기는 서버로 작업 요청을 하고 서버는 요청된 작업을 처리 후 단말로 그 결과를 전송 하여야 한다.

이 작업들은 주로 단말기가 서버인증모드(N/S 또는 NO)로 동작하고 있는 동안 발생하며, 서버는 인증 수행을 위하여 로컬 데이터베이스에 인증을 위한 사용자 정보를 저장 하고 있어야 한다.

또한 지문 인증을 수행하기 위해서는 지문 등록 및 인증 기능을 제공하는 UCB(UNION COMMUNITY Biometric) SDK를 사용할 필요가 있다.

사용자 인증타입 요청에 응답하기

단말기는 1:1 인증 절차를 수행하기 위하여 사용자의 인증 타입 정보가 필요하며 이를 획득 하기 위하여 서버로 UCSAPI_EVENT_REQUEST_VERIFYINFO 이벤트를 발생한다. 응용프로그램은 요청된 사용자의 이벤트 타입과 인증권한 정보를 을 단말기로 전송 한다. 단말기는 서버로부터 획득한 인증권한 정보가 인증가능(TRUE) 값을 가지면, 사용자의 지문 샘플(또는 비밀번호) 데이터를 채취하여 서버로 인증 요청을 하며, 그렇지 않으면 인증 절차를 인증 실패로 처리하여 종료 한다.

```
procedure EventRequestVerifyInfo(TerminalID: Integer; UserID: Integer);

// 이벤트에 대한 응답 방법
var
AuthType: Integer;
IsAccessibility: Integer;
ErrorCode: Integer;

AuthType = UCSAPI_AUTHTYPE_FP;   //사용자의 인증타입이 지문인경우.
IsAccessibility = 1; //인증 권한 있음. 인증권한이 없는 경우 0 이다.
ErrorCode = 0;   // 에러 없음
UCSCOMObj.ResponseVerifyInfo(TerminalID, UserID, AuthType, IsAccessibility, ErrorCode);
```

카드 인증 요청에 응답하기

단말기는 카드 인증을 수행하기 위하여 입력된 카드번호와 함께 서버로 UCSAPI_EVENT_REQUEST_CARDVERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 카드번보와 저장된 사용자의 카드번호를 비교 하여 그 결과를 단말기로 전송 한다.

```
procedure EventRequestCardVerifyMatch(TerminalID: Integer; AuthMode: Integer;
                                      const TextRFID: WideString);


// 이벤트에 대한 응답 방법
```

```
var
AuthType: Integer;
IsAccessibility: Integer;
IsMatched: Integer;
ErrorCode: Integer;
MatchedUserID: Integer;

AuthType = UCSAPI_AUTHTYPE_FP;  // 사용자의 인증타입이 지문인경우.
IsAccessibility = 1;  //인증 권한 있음. 인증권한이 없는 경우 0 이다.
IsMatched = 1;  //인증 수행 결과가 성공인 경우
ErrorCode = 0;  // 에러 없음
UCSCOMObj.ResponseCardVerifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched,
                                  ErrorCode);
```

1:1 인증 요청에 응답하기

단말기는 1:1 인증을 수행하기 위하여 입력된 지문 샘플(또는 비밀번호)과 함께 서버로 UCSAPI_EVENT_REQUEST_VERIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿을 비교 하여 그 결과를 단말기로 전송 한다.

```
procedure EventRequestVerifyMatch(TerminalID: Integer; UserID: Integer; AuthMode: Integer;
                                  SecuLevel: Integer; AuthType: Integer;
                                  const VerifyData: WideString);

// 이벤트에 대한 응답 방법
var
AuthType: Integer;
IsAccessibility: Integer;
IsMatched: Integer;
ErrorCode: Integer;
MatchedUserID: Integer;

AuthType = UCSAPI_AUTHTYPE_FP; // 사용자의 인증타입이 지문인경우.
IsAccessibility = 1; //인증 권한 있음. 인증권한이 없는 경우 0 이다.
IsMatched = 1; //인증 수행 결과가 성공인 경우
ErrorCode = 0; //에러 없음
UCSCOMObj.ResponseVerifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched,
ErrorCode);
```

1:N 지문 인증 요청에 응답하기

단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문과 함께 서버로 인증 요청을 하게 되고, 서버는 단말기로부터 획득한 지문 정보와 사용자들의 지문 정보를 비교하여 그 결과를 단말기로 회신하여야 한다.단말기는 1:N 지문 인증을 수행하기 위하여 입력된 지문 샘플과 함께 서버로

UCSAPI_EVENT_REQUEST_IDENTIFYMATCH 이벤트를 발생한다. 응용프로그램은 단말기로부터 획득한 지문샘플과 저장된 사용자의 지문 템플릿들을 비교 하여 그 결과를 단말기로 전송 한다.

```
procedure EventRequestIdentifyMatch(TerminalID: Integer; AuthMode: Integer;
                                    InputUserIDLength: Integer; SecuLevel: Integer;
                                    AuthType: Integer; const VerifyData: WideString);

// 이벤트에 대한 응답 방법
var
AuthType: Integer;
IsAccessibility: Integer;
IsMatched: Integer;
ErrorCode: Integer;
MatchedUserID: Integer;

AuthType = UCSAPI_AUTHTYPE_FP;  // 사용자의 인증타입이 지문인경우.
IsAccessibility = 1;  //인증 권한 있음. 인증권한이 없는 경우 0 이다.
IsMatched = 1;  //인증 수행 결과가 성공인 경우
ErrorCode = 0;  //에러 없음

//UserID는 인증 성공된 사용자 ID 이다.
UCSCOMObj.ResponseIdentifyMatch(TerminalID, MatchedUserID, IsAccessibility, IsMatched,
ErrorCode);
```

6. 단말기 관리 하기

서버에 접속중인 단말기 개수 얻기

응용프로그램은 서버에 접속중인 단말기의 개수를 얻기 위하여 GetTerminalCount() 메소드를 호출 한다.

```
UCSCOMObj.GetTerminalCount();

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;

//관련 이벤트
없음
```

단말기 펌웨어 버전 얻기

응용프로그램은 단말기의 펌웨어 버전을 얻기 위하여 GetFirmareVersion() 메소드를 호출 한다.

```
UCSCOMObj.GetFirmware(ClientID, TerminalID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;


//관련 이벤트
procedure EventFirmwareVersion(ClientID: Integer; TerminalID: Integer; const Version:WideString);
```

단말기 펌웨어 업그레이드 하기

응용프로그램은 단말기의 펌웨어를 업그레이드 하기 위하여 UpgradeFirmare() 함수를 호출 한다.

```
UCSCOMObj.GetFirmware(ClientID, TerminalID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;

//관련 이벤트 1 (업그레이드 진행중 이벤트)
procedure EventFirmwareUpgrading(ClientID: Integer; TerminalID: Integer; CurrentBlock: Integer;
                                 TotalBlock: Integer);
//관련 이벤트 2 (업그레이드 완료 이벤트)
procedure EventFirmwareUpgrade(ClientID: Integer; TerminalID: Integer);
```

단말기 옵션 값 얻기 및 설정 하기

UCSCOM 모듈은 단말기 옵션을 관리하기 위한 기능 들은 지원 하지 않는다.

단말기 잠금장치 강제 개방하기

응용프로그램은 단말기에 부착된 잠금 장치의 강제 개방을 위하여 OpenTerminal() 메소드를 호출한다.

```
UCSCOMObj.OpenTerminal(ClientID, TerminalID);

if UCSCOMObj.ErrorCode = UCSAPIERR_NONE Then
    begin
        //Success
    end;

//관련 이벤트
procedure EventOpenTerminal(ClientID: Integer; TerminalID: Integer);
```

8. 에러 핸들링(Error-Handling)

에러	코드	설명
UCSAPIERR_NONE	0x00000 0	에러가 없다
UCSAPIERR_NOT_ACTIVE	0x00000 1	서버가 실행 중이지 않다.
UCSAPIERR_MEMORY_ERROR	0x00000 2	메모리 에러
UCSAPIERR_INVALID_POINTER	0x00000 3	입/출력 함수 파라미터 또는 입/출력 구조체의 필드가 유효하지 않은 포인터 이다.
UCSAPIERR_INVALID_INPUT_POINTE R	0x00000 4	입력 파라미터 또는 입력된 구조체의 필드가 유효하지 않은 포인터 이다.
UCSAPIERR_INVALID_DATA	0x00000 5	입력된 파라미터가 유효하지 않다.
UCSAPIERR_INVALID_OUTPUT_POINT ER	0x00000 6	출력을 위한 파라미터 또는 구조체의 필드가 유효하지 않은 포인터 이다.
UCSAPIERR_OS_ACCESS_DENIED	0x00000 7	OS의 리소스에 접근에 제한 되었다.
UCSAPIERR_FUNCTION_FAILED	0x00000 8	함수가 알수 없는 이유로 실패 하였다.
UCSAPIERR_INCOMPATIBLE_VERSIO N	0x00000 9	버전 정보가 호환되지 않는다.
UCSAPIERR_INVALID_TERMINAL	0x00001 0	유효하지 않은 터미널 ID 이다
UCSAPIERR_START_SERVER	0x00001 1	서버 시작 실패.
UCSAPIERR_FW_FILE_NOTEXIST	0x00001 2	펌웨어 파일을 찾을 수 없다
UCSAPIERR_FW_DOWNLOAD	0x00001 3	펌웨 업그레이드 에러
UCSAPIERR_LOG_UPLOAD	0x00001	인증 기록 가져오기 실패

	4	
UCSAPIERR_TIMEOUT	0x000015	함수가 타임아웃의 이유로 실패 하였다.
UCSAPIERR_DATABASE	0x0007D3	데이터베이스 접근 에러
UCSAPIERR_INVALID_USER	0x000BBC	등록되지 않은 사용자 이다(서버 인증시 발생)
UCSAPIERR_MATCHING	0x000BBD	인증 실패(서버 인증시 발생)
UCSAPIERR_ACCESSIBILITY	0x000BBE	인증 권한 없음(서버 인증시 발생)
UCSAPIERR_ADDUSER	0x001389	단말기 사용자 추가 실패
UCSAPIERR_NOT_ENOUGH_MEMORY	0x00138A	단말기 메모리 부족
UCSAPIERR_WRITE_MEMORY	0x00138D	단말기 메모리 쓰기 에러(단말기가 이미 플레쉬 메모리에 쓰기중일 때 또 다른 쓰기 요청한 경우 발생)
UCSAPIERR_FIND_USER	0x00138E	서버에서 요청한 사용자를 찾을 수 없는 경우 발생
UCSAPIERR_FIRMWARE_VERSION	0x00138F	펌웨어 업그레이드중 버전이 맞지않아 업그레이드 할 수 없는 경우 발생
UCSAPIERR_UNKNOW_REASON	0x001770	알수 없는 에러
UCSAPIERR_CARD_DUPLICATED	0x001771	단말기에서 카드 등록시 카드번호 중복 발생