기술기준 적용 이해

1. ASN.1의 이해

1.1 ASN.1

○ ASN.1은 구조화된 정보를 기술하기 위한 언어로 추상 구문표기법(Abstract Syntax Notation One)의 약어임

- ASN.1은 OSI의 분야에 있어서 데이터형을 정의하기 위해 사용하는 기술 언어

○ BER과 DER은 ASN.1 형태의 추상 데이터 형식을 네트워크상에서 전송할 수 있는 형식으로 encoding해주고 그것을 다시 ASN.1 형태로 decoding해주는 방법에 대하여 정의함

- 네트워크 상에서 교환되는 메시지는 ASN.1형태로 구성되어 교환되지만, ASN.1 자체로는 추상적인 데이터 형식이므로 그것을 그대로 전송할 수 없음

1.2 ASN.1의 필요성

○ 각기 다른 시스템(예: UNIX와 WINDOWS)마다 데이터 표기법이 다르기 때문에 데이터를 전송해도 수신하는 쪽과 송신하는 쪽의 데이터가 다르게 해석될 수 있음

- 즉, Windows는 리틀엔디언(Little Endian)이어서 | 4 | 3 | 2 | 1 |, 오른쪽부터 해석되고, UNIX계열은 빅엔디언(Big Endian)이어서 | 1 | 2 | 3 | 4 |, 왼쪽부터 해석되는 방식이기 때문에 서로 다른 데이터로 인식함

예) 0x23을 32비트를 처리하는 빅엔디언에서는 (0x23 0x00 0x00 0x00)로 표기되지만 리틀엔디언에서는 (0x00 0x00 0x00 0x23)으로 표기

○ 또한 운영하는 장비가 8비트, 16비트, 32비트, 64비트 중 어느 것인지에 따라 변수의 크기가 달라지고, 송수신 시스템 사이에 동일한 언어를 사용하지 않고 다

른 언어를 사용한다면 더욱 복잡해지게 됨

예) 32비트 장비에서 long형은 4바이트이고, 64비트 장비에서 long형은 8바이트 임

○ 이러한 문제를 해결하기 위해서 공통화 된 데이터 표현문법을 만들게 되었고 이를 통해서 ASN.1이 제정되었음

2. ASN.1 범용툴을 이용한 기술기준 적용 방법

2.1 ASN.1 범용툴 개요

2.1.1 개발 배경

○ ITS 표준은 시스템 간 정보교환에 적용하기 위해 표준화된 데이터 표현방식을 정의할 필요가 있으며, 국토교통부에서 제정·고시한 기술기준은 국제표준(ISO)의 기술적 동향에 따라 플랫폼 및 운영체제에 독립적인 방법으로 정보를 표현할 수 있는 추상표기구문인 ASN.1(Abstract Syntax Notation One)을 적용함

○ 그러나 사업시행자는 ASN.1으로 정의된 기술기준을 적용하여 시스템을 개별적으로 구축하기에는 시간과 기술이 부족하고, 동일기술에 대한 중복투자로 경제적 손실이 발생함

○ 국외는 민간기업에서 기계언어로의 변환을 위한 상용툴을 개발하여 사용되고 있으나 라이센스 비용, 툴 유지비용 등으로 구축 및 유지보수에 과다 비용이 발생함

○ 이에 기술기준 적용의 용이성 및 교통정보 호환성 확보를 위해 정부차원에서 검증된 기술기준 구현 프로그램인 ASN.1 범용툴을 개발·제공하여 지원함

- ASN.1 범용툴 개발로 인해 기 구축된 시스템 또는 향후 구축될 시스템은 별도의 추가적인 투자비용 없이 보다 편리하게 기술기준 적용 가능

2.1.2 ASN.1 범용툴 기능

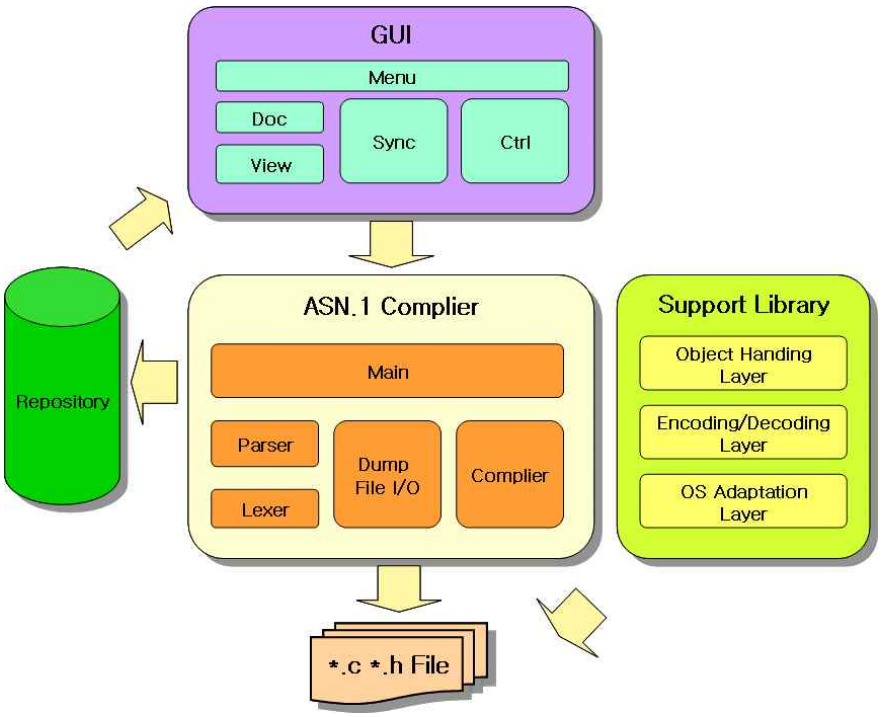
○ ASN.1 범용툴의 근본 기능은 ASN.1으로 정의된 기술기준을 기존의 개발환경에서 사용할 수 있도록 이를 변환해주는 것이며 기술기준을 Window 기반의 C 언어로 구현하기 위해 범용툴에서 제공하는 기능은 다음과 같음

〈표 1〉 ASN.1 범용툴 기능

기본 기능	내용
사용자 편의성 제공	‣ 사용의 편의성을 제공하기 위하여 GUI 기반의 사용자 인터페이스를 제공하며 사용자는 GUI 인터페이스를 통하여 ASN.1 정의파일 선택 및 컴파일 가능 ‣ ASN.1에 의하여 정의된 정보의 세부 정의 파악 가능
C언어 코드 생성	‣ ASN.1 표준에 의하여 정의된 기술기준을 시스템에 적용할 수 있도록 C 언어 기반의 코드 생성
전송구분 변환 기능	‣ 생성된 정보를 타 센터로 전송하기 위하여 ASN.1 구문을 전송구문으로 변환할 수 있는 기능 제공
시스템의 효율성 확보	‣ 기술기준 수정·변경 시 매번 기술기준 문서 전체에 대한 컴파일 작업을 수행함으로써 자원 낭비 ‣ 변경된 ASN.1 정의만을 컴파일 할 수 있는 기능을 추가하여 시스템의 효율성 확보 가능

2.1.3 ASN.1 범용툴 시스템 아키텍처

○ ASN.1 범용툴은 표준에 정의된 교통정보를 ITS 시스템에 적용될 수 있는 C언어 코드로 생성해주는 기능을 수행하며, 범용툴의 기능 블록은 ASN.1 컴파일러, Repository, Support Library, GUI 등 네 가지로 구성됨



〈그림 1〉 ASN.1 범용툴 시스템 아키텍처 구성도

(가) ASN.1 컴파일러

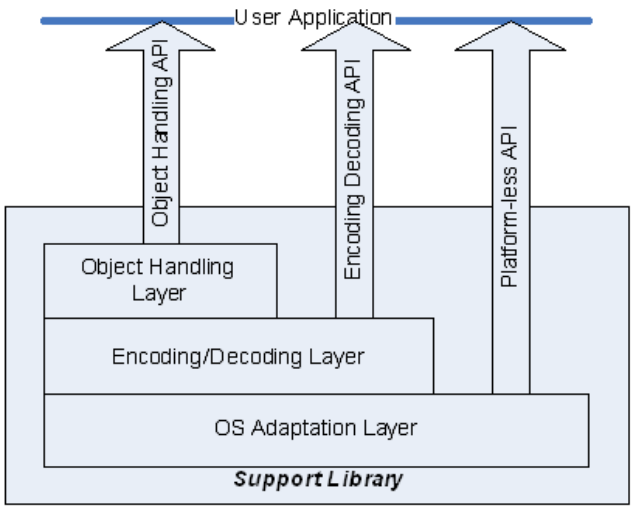
○ ASN.1 컴파일러는 다음의 기능블록으로 구성됨

- Main 블록 : ASN.1 문서의 문법검사, 소스 및 헤더파일 생성을 위한 관리기능을 수행
- Parser 블록 : ASN.1 문서의 문법 검사
- Lexer 블록 : 문법검사를 위한 토큰 생성
- Dump File I/O 블록 : Repository를 위한 Dump 파일 생성
- Complier 블록 : 소스 및 헤더파일 생성

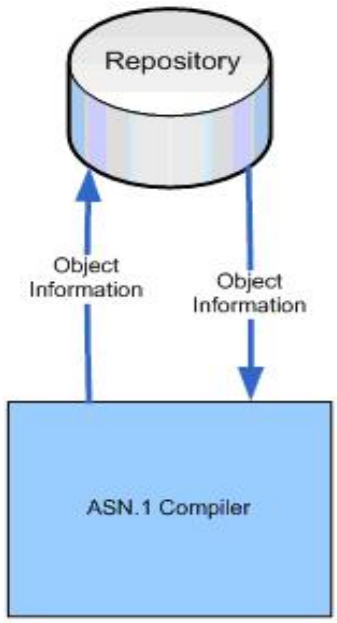
(나) Support Library

○ Support Library는 ASN.1 컴파일러에 의하여 생성된 C개발언어 소스 코드와 헤더파일이 정상적으로 동작할 수 있도록 지원함

- Object Handing Layer : 기능 호출 시 메모리 생성 및 삭제
- Encoding/Decoding Layer : Encoding/Decoding 기능 제공
- OS Adaptation Layer : Windows, Unix, Linux의 시스템 운영체제 지원을 위한 기능 제공



〈그림 2〉 Support Library Layer 구조



〈그림 3〉 Repository 구조도

(다) Repository

○ 기존 상용툴에는 없는 기능으로 물리적인 파일저장 공간인 Repository를 이용하여 컴파일 된 문서의 재 컴파일 수행 시 빠른 처리를 위한 기능을 수행할 수 있음

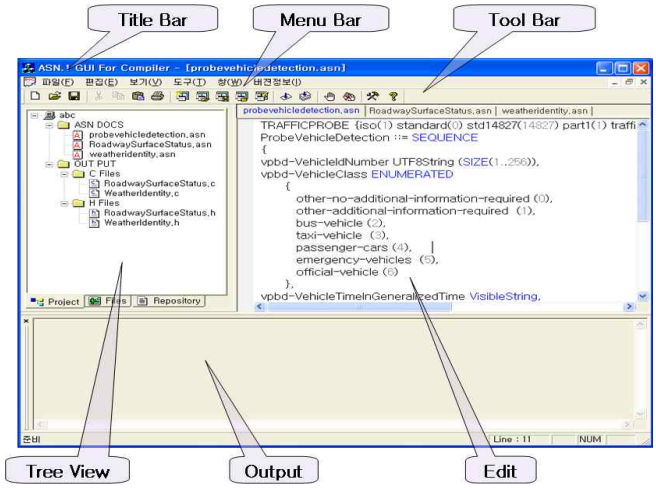
○ 사용자는 컴파일 된 문서의 자료구조를 Repository에 저장되어있는 Dump File을 이용하여 확인 할 수 있음

- GUI로 ASN.1 Repository에 저장된 ASN.1 문서의 구조 확인 기능 제공
- Repository 기능 제공을 위한 Dump File 저장
- Dump File 기능을 제공함으로써 재 컴파일 시 이미 만들어진 Dump File을 이용하여 즉각적인 소스 및 헤더파일 생성 기능 제공
- 컴파일 시간 단축

(라) GUI

○ GUI는 ASN.1 문서의 컴파일 시 편의기능을 제공하며 문서의 프로젝트 단위 관리 기능, 문서 편집 및 저장 등의 기능을 제공

- ASN.1 문서의 작성, 편집, 저장 기능
- 프로젝트 단위로의 생성, 저장 기능
- 트리뷰를 통한 ASN.1 문서 검색 기능
- 프로젝트뷰를 통한 프로젝트 단위의 관리 기능
- 단일 컴파일기능 및 프로젝트 단위의 컴파일 기능
- 컴파일 옵션 저장 기능
- 컴파일 된 ASN.1 문서의 자료구조를 보기 위한 Repository 화면 기능
- 문서편집기에서 제공되는 일반적인 파일 edit(저장, 편집, 삭제 등) 기능



〈그림 4〉 ASN.1 GUI 화면 구성

○ 위와 같은 기능을 수행하는데 사용되는 ASN.1 범용툴의 메뉴는 다음과 같음

〈표 2〉 범용툴 메뉴 기능

구분	메뉴	내용	메뉴	내용
파일	새 파일	새 문서	인쇄설정	프린터 설정
	열기	저장 문서 열기	새 프로젝트 생성	새 프로젝트 만들기
	닫기	열린 문서 닫기	프로젝트 열기	저장된 새 프로젝트 열기
	모두 닫기	열린 문서 모두 닫기	프로젝트 닫기	열린 새 프로젝트 닫기
	저장	작업 문서 저장하기	프로젝트 저장	작업 새 프로젝트 저장
	다른 이름으로 저장	다른 이름으로 저장하기	프로젝트 파일에 추가	프로젝트 파일에 추가하기
	인쇄	인쇄	종료	프로그램 종료
	인쇄 미리보기	인쇄 여백 설정		
편집	취소	입력 취소	다음 찾기	문장 중 다음 내용 검색
	잘라내기	작업 내용 잘라내기	바꾸기	입력한 내용으로 바꾸기
	복사	복사하기	북마크 표시/비표시	원하는 위치 표시 기능
	붙여넣기	복사 또는 잘라낸 내용 붙이기	북마크 삭제	표시한 부분 삭제
	찾기	입력한 내용 검색		
보기	폰트 변경	사용 폰트 변경	결과창	결과창 표시/비 표시
	트리뷰	트리뷰 표시/비 표시	도구모음	도구모음 표시/비 표시
도구	설정	컴파일 옵션 지정	프로젝트 컴파일	프로젝트 단위의 컴파일
	컴파일 실행	열린 문서의 컴파일	자동 줄 바꿈	자동 툴 바꿈
창	다음 창	다음 창 보기	바둑판 식	바둑판 모양으로 보기
	이전 창	이전 창 보기	아이콘 정렬	아이콘 정렬
	계단식	계단식으로 보기		
버전정보	버전정보	프로그램의 버전 정보		

2.2 ASN.1 범용툴 설치

2.2.1 배포 정책

○ 국토교통부에서는 기술기준을 적용하는 ITS 사업이나 ITS 표준 연구에 적극 활용될 수 있도록 ASN.1 범용툴의 배포 및 매뉴얼을 제공함

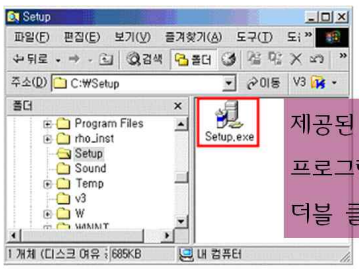
○ 기술기준 적용을 위해 개발된 목적으로만 사용되도록 기타의 개인 또는 단체의 영리를 위한 목적으로 무분별한 배포 및 복제를 방지하기 위하여 허가된 시스템 또는 사용자에게만 사용될 수 있도록 관리중임

* ASN.1 범용툴은 표준화전담기관을 통해 받을 수 있음

○ 국가의 ITS 관련 사업을 수행하는 경우 1년의 라이센스를 발급받아 ASN.1 범용툴 사용이 가능함

2.2.2 범용툴 설치과정

① 설치파일 실행



제공된 ASN.1 범용툴 설치 프로그램 Setup.exe. 파일을 더블 클릭

② 설치경로 지정

설치경로 지정 후 설치시작

③ 설치결과 확인

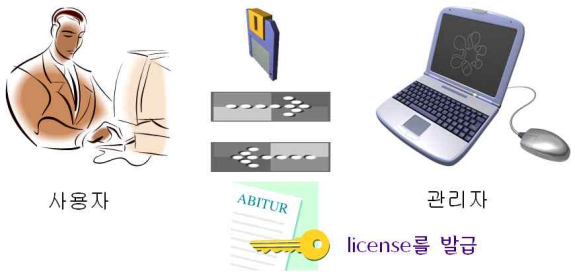
설치완료

바탕화면에 아이콘 생성

2.2.3 범용툴 라이센스 발급

① key.bin파일 생성

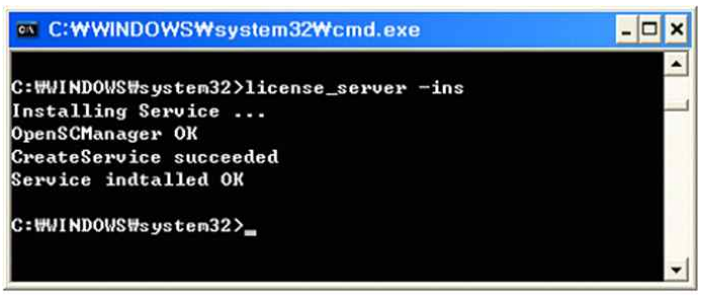
② 생성된 key.bin파일 관리자에게 전송 및 라이센스 수령



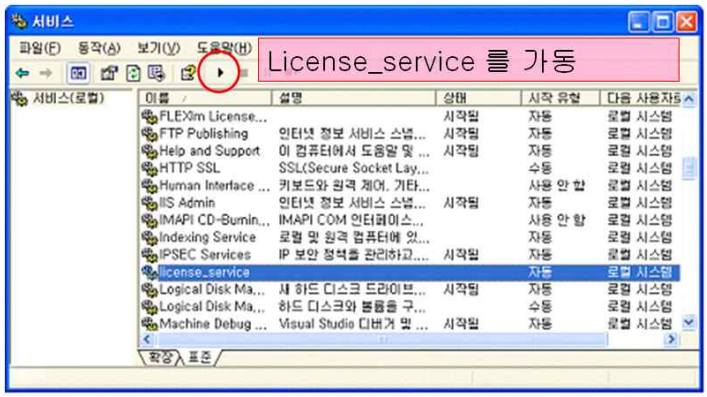
③ 라이센스 적용

④ 설치경로의 license_server파일을 System32(Win32)폴더로 이동

⑤ license_server 가동



⑥ 시작>제어판>관리도구>서비스 실행



2.3 ASN.1 범용툴 기본 사용방법

2.3.1 ASN.1 컴파일 생성

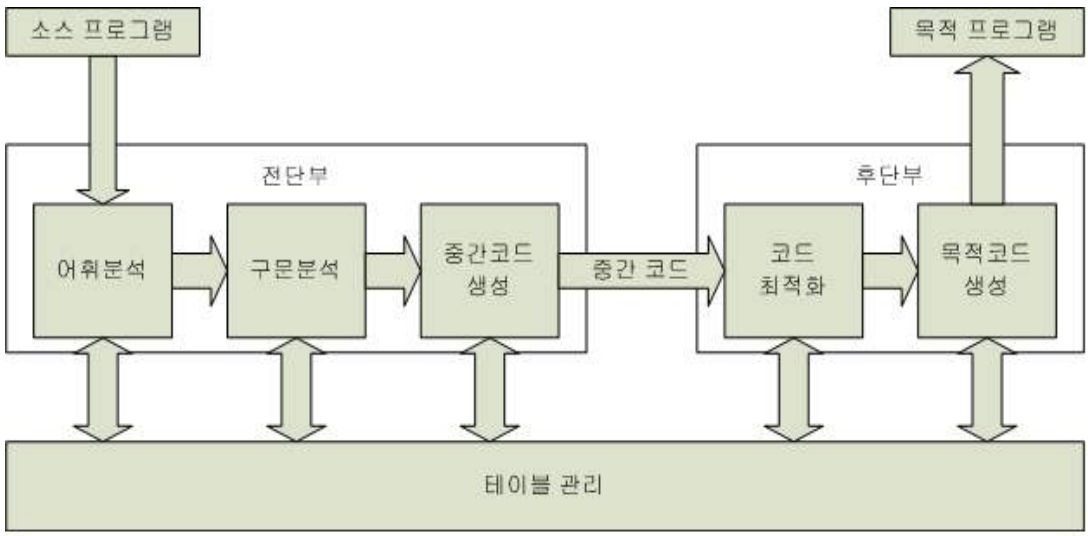
1) 컴파일러

○ 컴파일러는 일정한 문법을 가진 형식의 구문을 정해진 표준에 의해서 검사하여 실행 가능한 형태로 번역해주는 프로그램을 말하고 컴파일러 구조는 크게 전단부(Front-end)와 후단부(Back-end)로 나눌 수 있음

○ 전단부는 소스언어에 관계되는 부분으로써 소스 프로그램을 분석하고 어휘 분석 단계로부터 중간코드를 생성하는 역할을 담당함

○ 후단부는 전단부에서 생성된 중간코드를 특정 기계를 위한 목적코드로 번역하는 기능을 수행하여 최종적으로 목적 프로그램을 생성함

○ 일반적인 컴파일러의 구조는 다음과 같이 구성됨

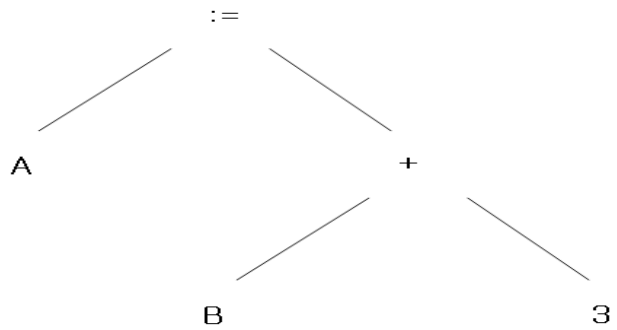


〈그림 5〉 컴파일러의 일반 구조

- 어휘분석 : 컴파일 과정의 첫 번째 단계로써 어휘분석기에 의해서 이루어지며 이는 소스 프로그램을 읽어 들여 일련의 토큰[1](Token)을 생성하도록 구성
- 구문분석 : 어휘분석 단계에서 생성된 토큰을 받아서 소스 프로그램에 대한 에러를 검사하고 올바른 문장에 대해서 구문구조 생성. 소스 프로그램에 에러가 있으면 그에 해당하는 에러 메시지를 출력하고 그렇지 않다면 프로그램에 대한 구문

1) 토큰이란 문법적 의미를 갖는 최소단위의 프로그램으로써 토큰의 열로 구성

구조를 트리 형태로 만들어 출력하도록 구성하고 트리 형태로는 파스트리와(Parser Tree)와 추상 구문 트리(Abstract syntax tree)가 있으며 근래 대부분의 컴파일러에서는 추상 구문 트리 사용



<그림 6> A := B+3;의 추상 구문 트리

- 중간코드 생성 : 파서의 출력인 추상 구문 트리의 입력을 받아서 의미 검사를 수행하고 그에 해당하는 중간 코드를 생성하는 것으로, 구문분석 단계에서 만들어진 구문구조를 이용하여 코드를 생성하고 구문 트리를 확인하여 각 구문에 대한 중간코드를 생성
- 코드 최적화 : 제공된 중간 코드를 의미적으로 동등하면서 좀 더 효율적인 코드로 변경하는 것으로 계산 횟수를 줄이고, 보다 빠른 명령을 사용하여 실행시간이 짧은 코드를 생성하며 메모리 사용을 최소화 하도록 구성
- 목적코드 생성 : 중간 코드를 입력받아서 그와 의미적으로 동등한 목적 기계에 대한 코드 생성

2) ASN.1 범용툴 컴파일 절차

○ ASN.1로 정의된 교통정보를 ASN.1 범용툴을 사용하여 컴파일 하는 과정은 아래 그림과 같은 절차에 의해서 이루어짐

① ASN.1에 의하여 정의된 기술기준을 ASN.1 GUI에서 또는 ASN.1 컴파일러 명령어 상에서 컴파일

② ASN.1 컴파일러 GUI를 이용하는 경우에, 사용자가 컴파일 수행 버튼을 누르면 ASN.1 정의 파일 이름, 필요 옵션(사용자 메뉴얼 참조)등과 함께 ASN.1 컴파일러에 전달

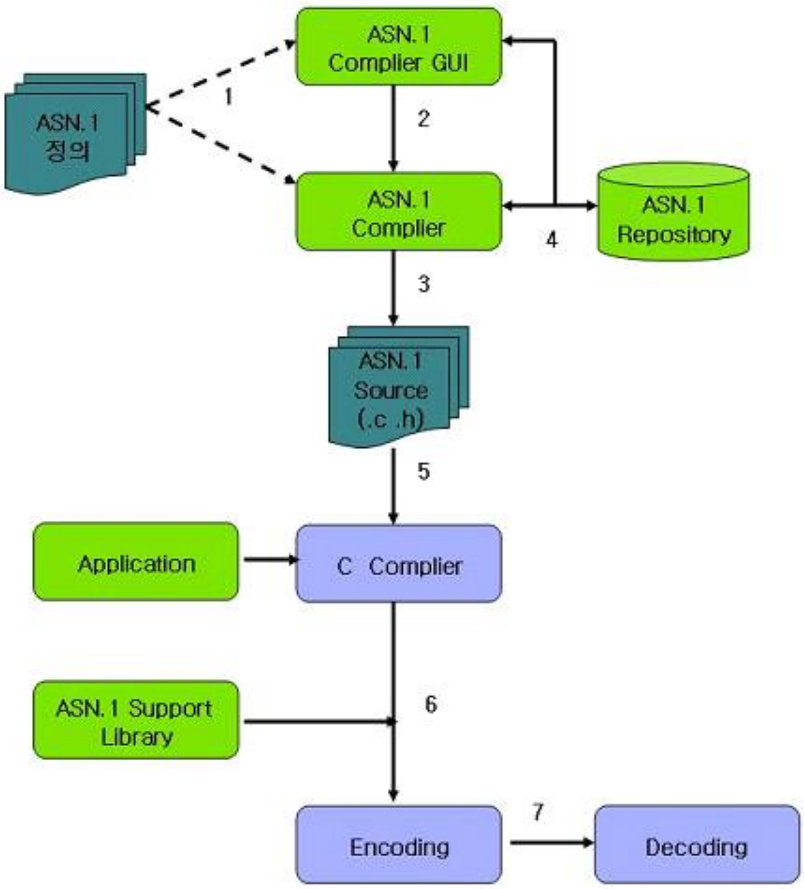
③ ASN.1 컴파일러는 ASN.1 정의 파일의 내용을 분석(Parsing)하고 유효 여부를 검사한 후, C언어 형태의 소스 및 헤더 파일 생성

④ ASN.1 컴파일러는 기술기준 내용을 분석 한 후 해당 ASN.1 정의를 GUI 또는 기타 다른 용도로 사용될 수 있는 형식으로 ASN.1 Repository에 저장

⑤ ASN.1 컴파일러로부터 생성된 소스 및 헤더파일을 기 개발된 Application과 연동

⑥ 소스 및 헤더파일을 Support Library의 지원 기능과 함께 C 컴파일러를 통해 컴파일하여 전송대상 시스템에 연결(Link)해서 사용할 수 있는 PDU 생성

⑦ 생성된 PDU를 전송 대상 시스템에 전송하며 전송받은 PDU를 Decoding하여 전송된 데이터 추출



〈그림 7〉 ASN.1 컴파일 과정

3) 런타임 라이브러리

○ Support Library는 ASN.1 컴파일러에 의하여 생성된 C개발언어 소스 코드와 헤더 파일이 정상적으로 동작할 수 있도록 하는 지원 기능을 제공함

- BER Encoding 규칙을 지원하는 Library 기능 제공
- 센터에 설치된 각종 OS를 지원하는 Library 기능 제공
- 기존 상용툴과의 연동기능 제공
- 센터-센터의 파일 전송을 위해 생성되는 PDU 대한 지원기능 제공

○ ASN.1 컴파일러에 의하여 생성된 C개발언어 소스 및 헤더파일이 정상적으로 동작할 수 있도록 하는 기본 지원 기능을 제공함

○ Support Library는 데이터타입의 생성, 삭제, 프린트, Encoding, Decoding의 기능을 제공하며, 이는 기능별로 각기 다른 역할을 담당하는 여러 함수로 구성됨

○ ASN.1에 의하여 정의된 모든 데이터에 관하여 다음과 같은 API 함수를 생성함

〈표 3〉 Support Library 생성 API

기본 API	설명
Encode_xxxx	C 언어 자료 구조에 저장된 데이터를 BER 네트워크로 전송할 수 있는 형태로 Encoding하는 API
Decode_xxxx	네트워크로부터 수신된 데이터를 BER로 Decoding하여 C 언어 프로그램에서 사용될 수 있는 C 언어 자료 구조로 저장하는 API
Print_xxxx	디버깅 또는 테스트를 위하여 C 언어 자료 구조에 저장된 데이터를 Print하는 API

2.3.2 Type Descriptor

1) 컴파일러 사용방법

○ File Name : ASN.1 범용 Support Library는 제공 기능별로 각기 다른 역할을 담당하는 여러 함수들로 구성되며, 이는 개념적인 의미임

○ 사용법 : asn1c [options ¦ filenames]

사용법
asn1c 컴파일러는 ITU-T X.680-0207 표준 규격에 명시한 Syntax로 작성된 문서를 Compile하며, 입력 파일은 4개의 스테이지에 따라 처리 : Parsing, Semantic Check, Dump, Compile
asn1c 컴파일러는 입력 파일의 확장자를 이용하여 파일의 타입 구분 *.asn : ASN.1 소스 파일 *.asn.dump : ASN.1 Object Dump 파일
asn1c 컴파일러는 입력 파일을 컴파일 하여 출력 파일 생성 *.c : C 소스 파일 *.h : C 헤더 파일
옵션(반드시 띄어 써야 함)
'-f'옵션과 -W옵션은 -f<name>과 같은 형태로 작성
l -E ASN.1문서에 대하여 Parsing만 수행하고 생성된 Object Tree를 화면에 출력
l -F -E 옵션과 같이 사용하며, Parsing후 구문 검사를 수행
l -P ASN.1 문서를 Parsing후 인식된 ASN.1 Text를 화면에 출력
l -X 화면 출력을 XML Format으로 출력
l -D ASN.1 문서를 Parsing후 생성된 Object Tree를 Dump File로 변환
l -Werror Compile시 발생 되는 경고를 에러로 취급
l -Wdebug_lexer Lexer Block에서 Debug Message를 출력
l -Wdebug_parser Patser Block에서 Debug Message를 출력
l -Wdebug_fixer 구문 검사를 처리하는 과정에서 Debug Message를 출력
l -Wdebug_compiler Compile Block에서 Debug Message를 출력
l -fall-defs-global C 소스에서 "asn1_DEF_*"로 생성되는 구조체 인스턴스를 전역 변수로 선언
l -fbless-SIZE C 소스에서 SIZE 제약 사항을 걸 수 없는 Data Type에도 SIZE제약 사항을 걸 수 있도록 허용
l -fcompound-names 생성되는 Structure의 이름을 모두 명확히 기술
l -findirect-choice CHOICE Object를 생성할 때, 값 대신 Pointer 참조
l -fknown-extern-type=<name> ASN.1 문서를 Parsing할 때 지정한 name을 이미 알고 있는 type으로 동작
l -fnative-types 변수 선언 시 가능한 INTEGER_t 대신 long 사용
l -fno-constraints Object에 대한 제약사항을 검사하는 코드 비생성

	-fno-include-deps #include 구문 비생성
	-funnamed-unions 구조체 내부에 이름이 지정되지 않은 union 사용 가능
	-ftypes88 ASN.1 문서를 Parsing할 때 1988년 규격 타입만 지원
	-print-constraint -E옵션을 사용하여 ASN.1 Object Tree를 출력 할 때, Constraint에 대한 주석 출력
	-print-lines -E옵션을 사용하여 ASN.1 Object Tree를 출력 할 때, ASN.1 Object가 선언된 ASN.1 문서의 줄 번호를 주석으로 출력
	-v asn1c의 Version 출력
	-h Help Message 출력

2) 런타임 라이브러리

○ Support Library는 기본적인 User API를 Type Descriptor를 통해서 제공함

○ OCTET STRING을 디코딩하고자 하는 경우, 사용자 Source Code에서 “asn_DEF_OCTET_STRING.ber_decoder”를 호출하며 Type Descriptor의 구조는 다음과 같음

Structure Name	asn_TYPE_descriptor_t	
Member		
Type	Name	Description
char *	name	Type의 이름
char *	xml_tag	XML Encode시 사용할 태그 명
asn_struct_free_f *	free_struct	해당 Type을 Free하는 함수의 주소
asn_struct_print_f *	print_struct	해당 Type의 내용을 출력하는 함수의 주소
asn_constr_check_f *	check_constraints	해당 Type의 constraint를 검사하는 함수의 주소
ber_type_decoder_f *	ber_decoder	주어진 Buffer를 BER에 의해 Decode하여 해당 Type을 만들어 내는 함수의 주소
der_type_encoder_f *	der_encoder	해당 Type을 DER에 의해 Encode하여 Data Stream을 만들어 내는 함수의 주소
다음 Member들은 내부적으로 처리하기 위해 정의하고, User Application에서 다음에 열거한 Member Function이나 Member 변수를 사용하여서는 안 됨		
asn_outmost_tag_f *	outmost_tag	해당 Type의 TAG를 추출하는 함수(Optional, Internal)
ber_tlv_tag_t *	tags	Sequence, Set, Choice에서 유효한 Member에 대한 Tag값을 저장하는 배열

int	tags_count	tags의 개수
ber_tlv_tag_t *	all_tags	Sequence, Set, Choice에서 모든 Member에 대한 Tag값을 저장하는 배열
int	all_tags_count	all_tags의 개수
asn_TYPE_member_s *	elements	Sequence, Set, Choice의 Member에 대한 정보를 저장하는 배열
int	elements_count	elements의 개수
void *	specifics	해당 Type만의 고유한 정보를 저장하고 있는 Structure의 주소

(가) free_struct

○ free_struct member Function은 메모리에 할당된 특정 Type Structure를 해제하는 함수임

Prototype	void free_struct(struct asn_TYPE_descriptor_s *type_descriptor, void *struct_ptr, int free_contents_only)
Description	메모리에 할당된 struct_ptr를 해제하는 것으로, free_contents_only가 0이 아닌 값으로 설정되면, struct_ptr의 내용만 Memory에서 해제하고 struct_ptr은 해제하지 않음
Return Value	없음
Arguments	
Name	Description
type_descriptor	해당 Type에 대한 정보를 담고 있는 Type Descriptor의 주소
struct_ptr	Memory에서 해제할 Data Buffer의 주소
free_contents_only	해제 방법을 설정하는 Flag
Example	

```
#include <asn_system.h>
#include <OCTET_STRING.h>
static int text_output(const void *buffer, size_t size,    void *param)
{
        unsigned int    i;
        FILE *fp =    (FILE *)param;
        unsigned char    *cp = (unsigned char *)buffer;
        for (i = 0; i    < size; i++) fputc((int)*cp++, fp);
        fputc('\n',    fp);
        return size;
}
int main(void)
{
        char *string    = "test string";
        OCTET_STRING_t    *octetString =
```

```
                        OCTET_STRING_new_fromBuf(&asn_DEF_OCTET_STRING,
                                                 string,
                                                 strlen(string));
        asn_DEF_OCTET_STRING.print_struct(&asn_DEF_OCTET_STRING,
                                          octetString,
                                          0,
                                          text_output, (void *)stdout);
        asn_DEF_OCTET_STRING.free_struct(&asn_DEF_OCTET_STRING,
                                         octetString,
                                         0);
        return 0;
}
```

(나) print_struct

○ print_struct은 해당 Type Structure의 내용을 사람이 읽을 수 있는 형태로 화면에 출력해 주는 함수임

Prototype	int print_struct(struct asn_TYPE_descriptor_s *type_descriptor, const void *struct_ptr, int level, asn_app_consume_bytes_f *callback, void *app_key)
Description	type_descriptor에 의거하여 struct_ptr의 내용을 사람이 읽을 수 있는 형태로 변환하여 화면에 출력
Return Value	성공 시 0을 실패 시 -1을 돌려줌
Arguments	
Name	Description
type_descriptor	해당 Type에 대한 정보를 담고 있는 Type Descriptor의 주소
struct_ptr	화면에 출력할 Data Buffer의 주소
level	들여쓰기 수준 값
call_back	사용자가 정의한 Data를 화면에 출력할 때 사용할 사용자 출력함수의 주소
app_key	call_back에 지정된 함수가 수행 하는데 필요한 인수를 저장한 Memory의 주소
Example	

```
#include    <asn_system.h>
#include    <OCTET_STRING.h>
static    int text_output(const void *buffer, size_t size, void *param)
{
        unsigned int i;
        FILE *fp = (FILE *)param;
        unsigned char *cp = (unsigned char    *)buffer;
        for (i = 0; i < size; i++)    fputc((int)*cp++, fp);     fputc('\n',    fp);
        return size;
}
int    main(void)
{
```

```
        char *string = "test    string";
        OCTET_STRING_t                          *octetString                    =
OCTET_STRING_new_fromBuf(&asn_DEF_OCTET_STRING,
                                                    string, strlen(string));
        asn_DEF_OCTET_STRING.print_struct(&asn_DEF_OCTET_STRING,    octetString,
                                        0, text_output, (void *)stdout);
        return 0;
}
```

(다) check_constraints

○ check_constraints는 해당 Type Structure의 내용이 ASN.1 문서에서 정의한 규칙에 위배되는 것이 없는지 검사해 주는 함수임

Prototype	int check_constraints (struct asn_TYPE_descriptor_s *type_descriptor, const void *struct_ptr, asn_app_consume_bytes_f *optional_app_errlog, void *optional_app_key);
Description	struct_ptr의 내용이 ASN.1 문서에서 정의한 규칙에 위배하는 것이 없는지 검사
Return Value	성공 시 0을 실패 시 -1을 돌려줌
Arguments	
Name	Description
type_descriptor	해당 Type에 대한 정보를 담고 있는 Type Descriptor의 주소
struct_ptr	규칙 검사를 수행할 Data Buffer의 주소
optional_app_errlog	에러 발생시 에러 메시지를 처리할 사용자 함수의 주소
optional_app_key	optional_app_errlog를 수행하는데 필요한 정보를 저장한 Memory의 주소

Example

```
#include    <asn_system.h>
#include    <OCTET_STRING.h>
static    int text_output(const void *buffer, size_t size, void *param)
{
        unsigned int i;
        FILE *fp = (FILE *)param;
        unsigned char *cp = (unsigned char    *)buffer;
        for (i = 0; i < size; i++)    fputc((int)*cp++, fp);    fputc('\n',    fp);
        return size;
}
int    main(void)
{
        char *string = "test    string";
        OCTET_STRING_t *octetString =
        OCTET_STRING_new_fromBuf(&asn_DEF_OCTET_STRING, string, strlen(string));
        asn_DEF_OCTET_STRING.check_constraints(&asn_DEF_OCTET_STRING,
                                                octetString,
                                                text_output, (void    *)stdout);
        return 0;
}
```

(라) ber_decoder

○ ber_decoder는 주어진 Buffer를 BER에 의해 디코딩하여 해당 Type의 구조를 생성해 주는 함수임

Prototype	asn_dec_rval_t ber_decoder(struct asn_codec_ctx_s *opt_codec_ctx, struct asn_TYPE_descriptor_s *type_descriptor, void **struct_ptr, const void *buf_ptr, size_t size, int tag_mode);
Description	buf_ptr에 주어진 Data를 Decoding하여 type_descriptor에 대응하는 Structure를 생성
Return Value	asn_dec_rval_t를 돌려줌
Arguments	
Name	Description
opt_codec_ctx	Decoding에 사용되는 Stack Memory의 양을 설정하는 Structure의 주소
type_descriptor	해당 Type에 대한 정보를 담고 있는 Type Descriptor의 주소
struct_ptr	생성된 Structure의 주소를 저장할 변수
buf_ptr	BER에 의해 Encoding된 Data가 저장되어 있는 Buffer의 주소
size	buf_ptr의 길이
tag_mode	Tag처리를 위한 Flag. -1인 경우 IMPLICT Tag Mode, 0인 경우 No Tag Mode, 1인 경우 EXPLICT Tag Mode로 동작
Example	

```
#include <asn_system.h>
#include <OCTET_STRING.h>
static unsigned char *load_file(char *fn, int *size)
{
        unsigned char   *buf = NULL;
        FILE *fp =   fopen(fn, "r+b");
        fseek(fp, 0,   SEEK_END); *size = ftell(fp); fseek(fp, 0, SEEK_SET);
        buf =   (unsigned char *)malloc(*size);
        fread(buf,   *size, 1, fp); fclose(fp);
        return buf;
}
int main(void)
{
        unsigned char   *buf = NULL;
        int size = 0;
        buf =   load_file("octetString.bin", &size);
        asn_DEF_OCTET_STRING.ber_decoder(NULL,&asn_DEF_OCTET_STRING,&result,buf,size,1);
        free(buf);
        return 0;
}
```

○ asn_dec_rval_t structure는 다음과 같음

Structure Name	asn_dec_rval_t	
Member		
Type	Name	Description
asn_dec_rval_code_e	code	결과 Code
size_t	consumed	Decoding에 사용된 Buffer의 Byte수

○ code에는 다음과 같은 값이 저장됨

Name	Value
RC_OK	0
RC_WMORE	1
RC_FAIL	2

○ Decoding후 code가 RC_WMORE인 경우 Decoding에 필요한 Data가 충분하지 않다는 것을 의미함

(마) der_encoder

○ der_encoder는 해당 Type의 구조를 주어진 DER에 의해 인코딩해 주는 함수임

Prototype	asn_enc_rval_t der_encoder(struct asn_TYPE_descriptor_s *type_descriptor, void *struct_ptr, int tag_mode, ber_tlv_tag_t tag, asn_app_consume_bytes_f *consume_bytes_cb, void *app_key)
Description	struct_ptr에 저장된 Data를 BER에 의해 Encdoing
Return Value	asn_enc_rval_t structure를 돌려줌
Arguments	
Name	Description
type_descriptor	해당 Type에 대한 정보를 담고 있는 Type Descriptor의 주소
struct_ptr	Encoding할 Data가 저장된 Data Buffer의 주소
tag_mode	Tag처리를 위한 Flag. -1인 경우 IMPLICT Tag Mode, 0인 경우 No Tag Mode, 1인 경우 EXPLICT Tag Mode로 동작
consume_bytes_cb	Encoding된 Data를 처리할 사용자 함수의 주소
app_key	consume_bytes_cb에 지정된 함수가 수행하는데 필요한 인수를 저장한 Memory의 주소
Example	

```
#include <asn_system.h>
#include <OCTET_STRING.h>
static int save_output(const void *buffer, size_t size,    void *param)
{
        FILE *fp =    (FILE *)param;
        fwrite(buffer,    size, 1, fp);
```

```
        return size;
}
int main(void)
{
        char *string    = "test string";
        asn_dec_rval_t    ret;
        OCTET_STRING_t                                          *octetString                =
OCTET_STRING_new_fromBuf(&asn_DEF_OCTET_STRING,
                                                string,    strlen(string));
        FILE *fp =    fopen("octetString.bin", "wb");
        asn_DEF_OCTET_STRING.der_encoder(&asn_DEF_OCTET_STRING,    octetString,
                                        1,    asn_DEF_OCTET_STRING.tags[0],
                                      save_output, (void    *)fp);
        fclose(fp);
        return 0;
}
```

(바) der_encoder function

○ asn_enc_rval_t는 다음과 같음

Structure Name	asn_enc_rval_t	
Member		
Type	Name	Description
size_t	encodded	Encoding된 Data의 크기로, -1인 경우 Encoding에 실패한 것을 의미
asn_TYPE_descriptor_s *	failed_type	Encoding에 실패한 Data의 Type Descriptor 주소
void *	structure_ptr	Encoding에 실패한 Data의 주소

○ Support Library가 기본적으로 제공하는 ASN Type에 대하여 Type Descriptor가 하나씩 존재하며, 각 ASN Type별 Type Descriptor의 이름과 이것을 정의한 헤더는 다음과 같음

ASN Type	Type Descriptor Name	Header File
ANY	asn_DEF_ANY	ANY.h
BIT_STRING	asn_DEF_BIT_STRING	BIT_STRING.h
BOOLEAN	asn_DEF_BOOLEAN	BOOLEAN.h
ENUMERATED	asn_DEF_ENUMERATED	ENUMERATED.h
GeneralString	asn_DEF_GeneralString	GeneralString.h
GeneralizedTime	asn_DEF_GeneralizedTime	GeneralizedTime.h
GraphicString	asn_DEF_GraphicString	GraphicString.h
IA5String	asn_DEF_IA5String	IA5String.h

ASN Type	Type Descriptor Name	Header File
INTEGER	asn_DEF_INTEGER	INTEGER.h
ISO646String	asn_DEF_ISO646String	ISO646String.h
NULL	asn_DEF_NULL	NULL.h
NativeEnumerated	asn_DEF_NativeEnumerated	NativeEnumerated.h
NativeInteger	asn_DEF_NativeInteger	NativeInteger.h
NativeReal	asn_DEF_NativeReal	NativeReal.h
NumericString	asn_DEF_NumericString	NumericString.h
OBJECT_IDENTIFIER	asn_DEF_OBJECT_IDENTIFIER	OBJECT_IDENTIFIER.h
OCTET_STRING	asn_DEF_OCTET_STRING	OCTET_STRING.h
ObjectDescriptor	asn_DEF_ObjectDescriptor	ObjectDescriptor.h
PrintableString	asn_DEF_PrintableString	PrintableString.h
REAL	asn_DEF_REAL	REAL.h
RELATIVE-OID	asn_DEF_RELATIVE-OID	RELATIVE-OID.h
T61String	asn_DEF_T61String	T61String.h
TeletexString	asn_DEF_TeletexString	TeletexString.h
UTCTime	asn_DEF_UTCTime	UTCTime.h
UTF8String	asn_DEF_UTF8String	UTF8String.h
UniversalString	asn_DEF_UniversalString	UniversalString.h
VideotexString	asn_DEF_VideotexString	VideotexString.h
VisibleString	asn_DEF_VisibleString	VisibleString.h

2.3.3 Support Type 및 보조 함수

1) Primitive Type

○ Primitive Type은 Type Descriptor를 가지지 않아 직접 사용하지 않고 ASN.1에서 정의하는 Primitive Class Data Type을 정의하기 위한 기본 구조임

○ Primitive Type 구조는 다음과 같음

Structure Name	ASN__PRIMITIVE_TYPE_t	
Member		
Type	Name	Description
uint8_t *	buf	Data가 저장되는 Buffer
int	size	buf의 크기

○ Primitive Type Data 구조의 상속 관계는 다음과 같음

```
Primitive Type – INTEGER – ENUMERATED
               – OBJECT_IDENTIFIER –  RELATIVE-OID
               – REAL
```

(가) INTEGER Type 보조 함수

Prototype	int asn_INTEGER2long(const INTEGER_t *i, long *l);
Description	INTEGER Type에 저장된 값을 long type 변수에 저장
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
i	INTEGER Type Structure의 주소
l	변환된 값이 저장될 Native Type 변수의 주소
Example	

```
int    main(void)
{
        INTEGER_t I;
        long initial = 100, data = 0;
        int status = 0;
        status = asn_long2INTEGER(&I,    initial);
        status = asn_INTEGER2long(&I,    &data);
}
```

Prototype	int asn_long2INTEGER(INTEGER_t *i, long l)
Description	long type에 저장된 값을 INTEGER Type Structure에 저장
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
i	변환된 값이 저장될 INTEGER Type Structure의 주소
l	값이 저장된 long type 변수
Example	

```
int    main(void)
{
        INTEGER_t I;
        long initial = 100, data = 0;
        int status = 0;
        status = asn_long2INTEGER(&I,    initial);
        status = asn_INTEGER2long(&I,    &data);
}
```

Prototype	const asn_INTEGER_enum_map_t *INTEGER_map_value2enum(asn_INTEGER_specifics_t *specs, long value)
Description	Enumerate로 정의한 INTEGER Type Structure의 정보를 이용하여, value에 해당하는 Enumerate 정보를 찾아냄
Return Value	해당 정보를 찾아내면 찾아낸 asn_INTEGER_enum_map_t structure의 주소를, 찾아내지 못하면 NULL을 돌려줌
Arguments	
Name	Description
specs	해당 Type에 대한 type descriptor의 specifics
value	Enumerate 정보를 찾을 숫자 값이 저장된 변수
Example	

```
int    main(void)
{
        long enum_val = 3;
        asn_INTEGER_enum_map_t *map =    INTEGER_map_value2enum(
                        (asn_INTEGER_specific_t *)asn_DEF_INTEGER.specific,    enum_val);
}
```

(나) OBJECT_IDENTIFIER 보조 함수

Prototype	int OBJECT_IDENTIFIER_get_arcs(OBJECT_IDENTIFIER_t *_oid, void *_arcs, unsigned int _arc_type_size, unsigned int _arc_slots)
Description	Object Identifier Structure에서 arc값을 얻어옴
Return Value	성공 시 얻어낸 arc의 개수를, 실패 시 -1을 돌려줌
Arguments	
Name	Description
_oid	arc를 얻어올 Object Identifier Structure의 주소
_arcs	arc를 저장할 배열의 주소
_arc_type_size	_arcs의 단위 크기(예 : sizeof(_arcs[0]))
_arc_slots	_arcs배열의 원소 개수
Example	

```
void    oid_example(OBJECT_IDENTIFIER_t *oid)
{
        long array[16];
        int slot_size = sizeof(array[0]), i;
        int status = OBJECT_IDENTIFIER_get_argc(oid, array,
                                                slot_size,    sizeof(array)/slot_size);
        for (i = 0; i < 16; i++)
                printf("%ld\n",    array[i]);
}
```

Prototype	int OBJECT_IDENTIFIER_set_arcs(OBJECT_IDENTIFIER_t *_oid, void *_arcs, unsigned int _arc_type_size, unsigned int _arc_slots)
Description	Object Identifier Structure에서 arc값 설정
Return Value	성공 시 설정한 arcs의 개수를, 실패 시 -1을 돌려줌
Arguments	
Name	Description
_oid	arc를 설정할 Object Identifier Structure의 주소
_arcs	설정할 arc를 저장하고 있는 배열
_arc_type_size	_arcs의 단위 크기(예 : sizeof(_arcs[0]))
_arc_slots	_arcs배열의 원소 개수
Example	

```
void    oid_example(OBJECT_IDENTIFIER_t *oid)
{
        long arcs[] = { 1, 3, 6, 1, 4, 1 };
        int slot_size = sizeof(arcs[0]);
        int status =    OBJECT_IDENTIFIER_set_arcs(oid, arcs,
                                                   slot_size,    sizeof(arcs)/slot_size)
}
```

Prototype	int OBJECT_IDENTIFIER_parse_arcs(const char *oid_text, ssize_t oid_txt_length, long arcs[], unsigned int arcs_slots, const char **opt_oid_text_end)
Description	Text로 작성되어 있는 OID를 해석하여 arcs배열에 저장
Return Value	성공 시 해석한 OID의 arc개수를, 실패 시 -1을 돌려줌
Arguments	
Name	Description
oid_text	Text로 작성된 OID가 저장된 Buffer의 주소
oid_txt_length	oid_text의 길이
arcs[]	해석된 OID의 arc값이 저장될 배열
arcs_slots	arcs배열의 크기
opt_oid_text_end	해석 종료 후, 남은 Text의 시작 위치
Example	

```
void    oid_example(void)
{
        char *text = "1.3.6.1.4.1";
        long arcs[16] = { 0, };
        int i;
        int cnt =    OBJECT_IDENTIFIER_parse_arcs(text, -1, arcs, 16, NULL);
        for (i=0; i<cnt; i++)
                printf("%ld\n",    arcs[i]);
}
```

Prototype	int RELATIVE_OID_get_arcs(RELATIVE_OID_t *_roid, void *arcs, unsigned int arc_type_size, unsigned int arc_slots)
Description	Relative OID Structure에서 arc값을 얻어옴
Return Value	성공 시 얻어낸 arc의 개수를, 실패 시 -1을 돌려줌
Arguments	
Name	Description
_roid	arc를 얻어올 Relative OID Structure의 주소
arcs	arc값을 저장할 배열
arc_type_size	arc배열 원소의 크기
arc_slots	arc배열 원소의 개수

Prototype	int RELATIVE_OID_set_arcs(RELATIVE_OID_t *_roid, void *arcs, unsigned int arc_type_size, unsigned int arcs_slots)
Description	Relative OID Structure에 arc값 설정
Return Value	성공 시 설정한 arc의 개수를, 실패 시 -1을 돌려줌
Arguments	
Name	Description
_roid	arc값을 설정할 Relative OID Structure의 주소
arcs	설정할 값을 저장하고 있는 arc배열
arc_type_size	arc배열 원소의 크기
arcs_slots	arc배열 원소의 개수

Prototype	int asn_REAL2double(const REAL_t *real_ptr, double *d)
Description	REAL Structure에 저장된 정보를 double 변수로 변환
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
real_ptr	변환할 REAL Structure의 주소
d	변환된 값이 저장될 변수의 주소

Prototype	int asn_double2REAL(REAL_t *real_ptr, double d)
Description	double 변수에 저장된 값을 변환하여 REAL Structure에 저장
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
real_ptr	변환된 값을 저장할 REAL Structure의 주소
d	변환할 값을 저장한 Double변수

2) Any Type

○ Any Type은 모든 Type을 수용 가능하며 그 구조는 다음과 같음

Structure Name	ANY_t	
Member		
Type	Name	Description
uint8_t *	buf	data가 저장되는 Buffer
int	size	buf의 크기
asn_struct_ctx_t	_asn_ctx	Data를 조작하기 위한 부가 정보

(가) Any_fromType

Prototype	int ANY_fromType(ANY_t *from, asn_TYPE_descriptor_t *td, void *struct_ptr)
Description	struct_ptr에 지정된 내용을 type descripter의 정보를 이용하여 ANY Type Structure에 저장
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
from	변환된 Data가 저장될 ANY Structure의 주소
td	변환할 Data에 대한 Type Descriptor의 주소
struct_ptr	변환될 Data가 저장된 Structure의 주소
Example	

```
void    any_example(void)
{
        char *string = “test_string”;
        ANY_t from;
        OCTET_STRING_t *octetString = OCTET_STRING_new_fromBuf(
                                &asn_DEF_OCTET_STRING,    string, strlen(string));
        ANY_fromType(&from, &asn_DEF_OCTET_STRING,    octetString);
        return ret;
}
```

(나) ANY_new_fromType

Prototype	ANY_t *ANY_new_fromType(asn_TYPE_descriptor_t *td, void *struct_ptr)
Description	struct_ptr에 지정된 내용을 type descripter의 정보를 이용하여 ANY Type Structure로 생성
Return Value	성공 시 생성된 ANY Structure의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
td	변환할 Data에 대한 Type Descriptor의 주소

struct_ptr	변환할 Data를 저장한 Structure의 주소
Example	

```
ANY_t    *octetString2Any(OCTET_STRING_t *octetString)
{
        ANY_t *ret = ANY_new_fromType(&asn_DEF_OCTET_STRING,    octetString);
        return ret;
}
```

(다) ANY_to_type

Prototype	int ANY_to_type(ANY_t *to, asn_TYPE_descriptor_t *td, void **struct_ptr)
Description	ANY Structure에 저장된 정보를 Type Descriptor를 이용하여 변환한 다음 해당 Type의 Structure를 생성하고 변환된 Data를 생성한 Structure에 저장
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
to	변환할 Data를 저장한 ANY Structure의 주소
td	변환될 Data에 대한 Type Descriptor의 주소
struct_ptr	생성된 Structure의 주소가 저장될 변수
Example	

```
OCTET_STRING_t    *any2OctetString(ANY_t *any)
{
        OCTET_STRING_t *octetString = NULL;
        ANY_to_type(any, &asn_DEF_OCTET_STRING,    &octetString);
        return octetString;
}
```

(라) ANY_fromBuf

Prototype	ANY_fromBuf(s, buf, size)
Description	주어진 Buffer를 이용하여 ANY Structure를 설정하고 이 매크로는 ANY_fromType 함수 호출
Arguments	
Name	Description
s	ANY Structure의 주소
buf	Buffer의 주소
size	Buffer의 길이
Example	

```
void    any_example(void)
{
        ANY_t any;
        unsigned char buf[128] = { 0xff, };
        ANY_fromBuff(&any, buf, 128);
}
```

(마) ANY_new_fromBuf

Prototype	ANY_new_fromBuf(buf, size)
Description	주어진 Buffer를 이용하여 ANY Structure를 생성하고 이 매크로로 ANY_new_fromType 함수 호출
Arguments	
Name	Description
buf	Buffer의 주소
size	Buffer의 길이
Example	

```
void    any_example(void)
{
        unsigned char buf[128] = { 0xff, };
        ANY_t *any = ANY_new_fromBuff(buf,    128);
}
```

3) BIT_STRING Type

○ BIT_STRING Type은 Binary Data를 저장하는 Type이며 그 구조는 다음과 같음

Structure Name	BIT_STRING_t	
Member		
Type	Name	Description
uint8_t *	buf	data가 저장되는 Buffer
int	size	buf의 크기
int	bits_unused	buf의 마지막 Byte에서 사용하지 않는 Bit의 수
asn_struct_ctx_t	_asn_ctx	Data를 조작하기 위한 부가 정보

4) OCTET_STRING Type

○ OCTET_STRING Type은 문자열 Data를 저장하는 Type이고 그 구조는 다음과 같음

Structure Name	OCTET_STRING_t	
Member		
Type	Name	Description
uint8_t *	buf	data가 저장되는 Buffer
int	size	buf의 크기
asn_struct_ctx_t	_asn_ctx	Data를 조작하기 위한 부가 정보

(가) OCTET_STRING_fromBuf

Prototype	int OCTET_STRING_fromBuf(OCTET_STRING_t *s, const char *str, int size)
Description	주어진 Buffer를 이용하여 Octet String Structure를 설정
Return Value	성공 시 0을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
s	값을 설정 할 Octet String Structure의 주소
*str	buffer의 주소
size	buffer의 길이
Example	

```
void    octet_string_example(void)
{
         unsigned char buf[128] = { 0xff, };
         OCTET_STRING_t octetString;
         OCTET_STRING_fromBuff(&octetString,    buf, 128);
}
```

(나) OCTET_STRING_new_fromBuf

Prototype	OCTET_STRING_t *OCTET_STRING_new_fromBuf(asn_TYPE_descriptor_t *td, const char *str, int size)
Description	주어진 Buffer를 이용하여 Octet String Structure를 생성
Return Value	성공 시 생성된 Octet String Structure의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
td	OCTET STRINT Type에 대한 Type Descriptor의 주소
str	Buffer의 주소
size	Buffer의 길이
Example	

```
void    octet_string_example(void)
{
         unsigned char buf[128] = { 0xff, };
         OCTET_STRING_t *octetString =    OCTET_STRING_fromBuff(buf, 128);
}
```

(다) codeconv_open

Prototype	codeconv_t codeconv_open(char *fromcs, char *tocs)
Description	fromcs에 주어진 Code Set을 tocs에 주어진 Code Set으로 변환하는 Code Converting Handle을 Open
Return Value	성공 시 생성된 Code Converter Handle의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
fromcs	원본 Text의 Code Set Name
tocs	변환하고자 하는 Code Set Name
Example	

```
void    utf8String_example(void)
{
        codeconv_t handle = codeconv_open("UTF8",    "EUCKR");
        char *nativeStr = "한글KSC5601", *nsp = nativeStr;
        char *utf8Str[128], *usp = utf8Str;
        size_t size = codeconv(handle,    &nsp, NULL, &usp, NULL);
        codeconv_close(handle);
}
```

(라) codeconv

Prototype	size_t codeconv(codeconv_t cd, char **inbuf, size_t *inbytesleft, char **outbuf, size_t *outbytesleft)
Description	Code Converter Handle을 이용하여 문자열 변환
Return Value	변환한 문자열의 개수를 돌려줌
Arguments	
Name	Description
cd	Code Converter Handle
inbuf	변환할 문자가 저장된 buffer pointer의 주소로 함수가 종료되면 다음 변환을 시작할 위치 지시
inbytesleft	함수가 종료되면 inbuf의 정보를 사용하고 남은 바이트 수 저장
outbuf	변환된 문자가 저장될 buffer pointer의 주소와 함수가 종료되면 다음 변환된 문자가 저장될 위치 지시
outbytesleft	함수가 종료되면 outbuf를 사용하고 남은 바이트 수 저장
Example	

```
void    utf8String_example(void)
{
        codeconv_t handle = codeconv_open("UTF8",    "EUCKR");
        char *nativeStr = "한글KSC5601", *nsp = nativeStr;
        char *utf8Str[128], *usp = utf8Str;
        size_t size = codeconv(handle,    &nsp, NULL, &usp, NULL);
        codeconv_close(handle);
}
```

(마) codeconv_close

Prototype	int codeconv_closeonv_t cd)
Description	Code Converter Handle 닫음
Return Value	항상 0을 리턴
Arguments	
Name	Description
cd	Code Converter Handle
Example	

```
void    utf8String_example(void)
{
        codeconv_t handle = codeconv_open("UTF8",    "EUCKR");
        char *nativeStr = "한글KSC5601", *nsp = nativeStr;
        char *utf8Str[128], *usp = utf8Str;
        size_t size = codeconv(handle,    &nsp, NULL, &usp, NULL);
        codeconv_close(handle);
}
```

(바) asn_strings_get_length

Prototype	asn_strings_get_length(x)
Description	OCTET STRING Type 또는 OCTET STRINT Type에서 생성된 모든 String 계열의 Type에 대한 String의 길이 추출
Arguments	
Name	Description
x	OCTET STRING계열의 Structure주소
Example	

```
void    octet_string_example(void)
{
        unsigned char buffer[128] = { 0xff, }, *buf = NULL;
        OCTET_STRING_t *octetString =    OCTET_STRING_new_fromBuff(buffer, 128);
        unsigned char *buf =    malloc(asn_strings_get_length(octetString));
}
```

(사) OCTET-STRING_fromString

Prototype	OCTET_STRING_fromString(s, str)
Description	String을 이용하여 OCTET STRING Structure를 설정하고 이 매크로로 OCTET_STRING_fromBuf 호출
Arguments	
Name	Description
s	String을 설정할 OCTET STRING Structure의 주소
str	문자열이 저장된 Buffer의 주소

Example

void octet_string_example(void)
{
unsigned char *str = “test string”;
OCTET_STRING_t octetString;
OCTET_STRING_fromString(&octetString, str);
}

5) GeneralizedTime Type

(가) asn_GT2time

Prototype	time_t asn_GT2time(const GeneralizedTime_t *gt, struct tm *_optional_tm4fill, int as_gmt)
Description	GeneralizedTime에 저장된 값을 time_t로 변환
Return Value	성공 시 변환된 초 값을, 실패 시 –1을 돌려줌
Arguments	
Name	Description
gt	변환할 값을 저장하고 있는 GeneralizedTime Structure의 주소
_optionl_tm4fill	변환된 값을 저장할 tm structure의 주소(Optional)
as_gmt	_optionl_tm4fill structure에 저장할 시간을 local time으로 저장할지, GMT Time으로 저장할지 결정하는 flag. (_optionl_tm4fill이 NULL이 아닌 경우에만 유효)
Example	

void GT_example(GeneralizedTime_t *gt)
{
time_t jiffie = asn_GT2time(gt, NULL, 0);
printf(“time = %ld\n”, jiffie);
}	

(나) asn_GT2time_frac

Prototype	time_t asn_GT2time_frac(const GeneralizedTime_t *tp, int *frac_value, int *frac_digits, truct tm *_optional_tm4fill, int as_gmt);
Description	GeneralizedTime에 저장된 값을 time_t로 변환하는 것으로, frac_value 및 frac_digits 값이 주어지면 1초 이하의 값을 변환하여 frac_value 및 frac_digit에 저장
Return Value	성공 시 변환된 초 값을, 실패 시 –1을 돌려줌
Arguments	
Name	Description
tp	변환할 값을 저장하고 있는 GeneralizedTime Structure의 주소
frac_value	1초 이하의 시간 값이 저장될 변수의 주소
frac_digits	1초 이하의 소수점 자릿수

_optional_tm4fill	변환된 값을 저장할 tm structure의 주소(Optional)
as_gmt	_optionl_tm4fill structure에 저장할 시간을 local time으로 저장할지, GMT Time으로 저장할지 결정하는 flag. (_optionl_tm4fill이 NULL이 아닌 경우에만 유효)

Example

```
void    GT_example(GeneralizedTime_t *gt)
{
        int value = 0, digit = 0;
        time_t jiffie = asn_GT2time_frac(gt,    &value, &digit, NULL, 0);
        double t = (double)jiffie + value * (1    / pow(10, digit));
        printf("time = %lf\n", t);
}
```

(다) asn_GT2time_prec

Prototype	time_t asn_GT2time_prec(const GeneralizedTime_t *tp, int *frac_value, int frac_digits, struct tm *_optional_tm4fill, int as_gmt)
Description	GeneralizedTime에 저장된 값을 time_t로 변환하는 것으로, frac_value 및 frac_digits가 값이 주어지면 1초 이하의 값을 변환하여 frac_value 및 frac_digit에 저장
Return Value	성공 시 변환된 초 값을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
tp	변환할 값을 저장하고 있는 GeneralizedTime Structure의 주소
frac_value	1초 이하의 시간 값이 저장될 변수의 주소
frac_digits	1초 이하의 소수점 자릿수
_optional_tm4fill	변환된 값을 저장할 tm structure의 주소(Optional)
As_gmt	_optionl_tm4fill structure에 저장할 시간을 local time으로 저장할지, GMT Time으로 저장할지 결정하는 flag. (_optionl_tm4fill이 NULL이 아닌 경우에만 유효)

Example

```
void    GT_example(GeneralizedTime_t *gt)
{
        int value = 0, digit = 3;
        time_t jiffie = asn_GT2time_prec(gt,    &value, digit, NULL, 0);
        double t = (double)jiffie + value * (1    / pow(10, digit));
        printf("time = %.3lf\n", t);
}
```

(라) asn_time2GT

Prototype	GeneralizedTime_t *asn_time2GT(GeneralizedTime_t *_optional_gt, const struct tm *tm, int force_gmt)
Description	tm structure의 내용을 변환하여 GeneralizedTime에 설정하고 _optional_gt가 NULL이면 GeneralizedTime 생성

Return Value	성공 시 설정된(혹은 생성된) GeneralizedTime Structure의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
_ optional_gt	변환된 시간 값이 저장될 GeneralizedTime Structure의 주소(Optional)
tm	변환할 시간 값이 저장된 tm Structure의 주소
force_gmt	설정(혹은 생성)되는 GeneralizedTime Structure를 GMT로 설정하도록 하는 Flag
Example	

```
void    GT_example(void)
{
        time_t now = time(NULL);
        struct tm *tmp = localtime(&now);
        struct tm tm = *tmp;
        GeneralizedTime_t *gt =    asn_time2GT(NULL, &tm, 0);
}
```

(마) asn_time2GT_frac

Prototype	GeneralizedTime_t *asn_time2GT_frac(GeneralizedTime_t *_optional_gt, const struct tm *tm, int frac_value, int frac_digits, int force_gmt);
Description	tm structure의 내용을 변환하여 GeneralizedTime에 설정하고 _optional_gt가 NULL이면 GeneralizedTime 생성
Return Value	성공시 설정된(혹은 생성된) GeneralizedTime Structure의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
_optional_gt	변환된 시간 값이 저장될 GeneralizedTime Structure의 주소(Optional)
tm	변환할 시간 값이 저장된 tm Structure의 주소
frac_value	1초 이하의 시간 값
frac_digits	1초 이하 시간 값의 소수점 자릿수
force_gmt	설정(혹은 생성)되는 GeneralizedTime Structure을 GMT로 설정하도록 하는 Flag
Example	

```
void    GT_example(void)
{
        struct timeval tNow = { 0, 0 };
        struct tm *tmp = NULL, tm;
        GeneralizedTime_t *gt;

        gettimeofday(&tNow, NULL);
        tmp = localtime(&tNow.tv_sec);
        tm = *tmp;
        gt = asn_time2GT_frec(NULL, &tm,    tNow.tv_nsec, 9, 0);
}
```

(바) asn_UT2time

Prototype	time_t asn_UT2time(const UTCTime_t *tp, struct tm *_optional_tm4fill, int as_gmt);
Description	UTCTime에 저장된 값을 time_t로 변환
Return Value	성공 시 변환된 초 값을, 실패 시 -1을 돌려줌
Arguments	
Name	Description
tp	시간 값이 저장된 UTCTime Structure의 주소
_optional_tm4fill	변환된 값을 저장할 tm structure의 주소(Optional)
as_gmt	_optionl_tm4fill structure에 저장할 시간을 local time으로 저장할지, GMT Time으로 저장할지 결정하는 flag. (_optionl_tm4fill이 NULL이 아닌 경우에만 유효)
Example	

```
void    UT_example(UTCTime_t *ut)
{
        time_t jiffie = asn_UT2time(ut, NULL,    0);
        printf("time = %ld\n", jiffie);
}
```

(사) asn_time2UT

Prototype	UTCTime_t *asn_time2UT(UTCTime_t *__opt_ut, const struct tm *tm, int force_gmt);
Description	tm structure의 내용을 변환하여 UTCTime에 설정하고 _optional_gt가 NULL이면 UTCTime을 생성
Return Value	성공시 설정된(혹은 생성된) UTCTime Structure의 주소를, 실패 시 NULL을 돌려줌
Arguments	
Name	Description
__opt_ut	변환된 값을 저장할 UTCTime Structure의 주소
tm	변환할 시간 값이 저장된 tm Structure의 주소
force_gmt	설정(혹은 생성)되는 UTCTime Structure를 GMT로 설정하는 Flag
Example	

```
void    UT_example(void)
{
        time_t now = time(NULL);
        struct tm *tmp = localtime(&now);
        struct tm tm = *tmp;
        UTCTime_t *ut = asn_time2UT(NULL,    &tm, 0);
}
```

6) UTF8String Type

(가) UTF8String_length

Prototype	ssize_t UTF8String_length(const UTF8String_t *st);
Description	UTF8String에 저장된 문자열이 WCS로 변환되었을 때의 길이 계산
Return Value	성공 시 문자열의 길이를, 실패 시 다음과 같은 값을 돌려줌 - 1 : UTF8 시퀀스가 잘림 - 2 : UTF8 시퀀스 시작 값이 틀림 - 3 : 연속 예외 처리 실패 - 4 : 처리하기 위한 최소 길이보다 Data가 적음 - 5 : 잘못된 인수
Arguments	
Name	Description
st	문자열 길이를 계산할 정보가 저장된 UTF8String Structure의 주소
Example	

```
void    utf8String_example(void)
{
        UTF8String_t *utf8 = NULL;
        codeconv_t handle = codeconv_open("UTF8",    "EUCKR");
        char *nativeStr = "한글KSC5601";
        char *utf8Str[128];
        size_t size = codeconv(handle,    nativeStr, NULL, utf8Str, NULL);
        ssize_t test = 0;
        codeconv_close(handle);
        utf8 = (UTF8String_t    *)OCTET_STRING_new_fromBuff(utf8Str, size);
        test = UTF8String_length(utf8);
        printf("Native = %d, UTF8 = %d\n",    strlen(nativeStr), test);
}
```

(나) UTF8String_to_wcs

Prototype	size_t UTF8String_to_wcs(const UTF8String_t *st, uint32_t *dst, size_t dstlen);
Description	UTF8String Structure에 저장된 Data를 WCS 문자열로 변환
Return Value	UTF8String_length의 Return Value 참조
Arguments	
Name	Description
st	변환할 Data가 저장된 UTF8String Structure의 주소
dst	변환된 Data가 저장될 Buffer의 주소
dstlen	Buffer의 길이
Example	

```
void    utf8String_example(UTF8String *st)
{
        ssize_t size = UTF8String_length(st);
        char *buf = malloc(size + 1);
        UTF8String_to_wcs(st, buf, size);
}
```

2.3.4 범용툴을 이용한 컴파일 및 예제

1) ASN.1 Document

○ Rectangle이라는 Type을 다음과 같이 정의함

```
RectangleModule DEFINITION ::= BEGIN
  Rectangle ::= SEQUENCE {
            height        INTEGER,
            width         INTEGER
    }
END
```

2) 컴파일

○ 상기에서 작성한 Rectangle.asn을 컴파일하여 Rectangle.h와 Rectangle.c를 얻음

```
c:\asn1c> asn1c -fnative-types Rectangle.asn1
```

3) 코드구성

○ 다음 Source는 Rectangle.h에 정의된 asn_DEF_Rectangle Type Descriptor를 이용하여, Rectangle_t Type의 구조를 인코딩함

```
#include <stdio.h>
#include <sys/types.h>
#include <Rectangle.h>    /* Rectangle ASN.1 type  */

static int
write_out(const void *buffer, size_t size, void *app_key) {
    FILE *out_fp = app_key;
    size_t wrote;

    wrote = fwrite(buffer, 1, size, out_fp);

    return (wrote == size) ? 0 : -1;
}

int main(int ac, char **av) {
    Rectangle_t *rectangle;               /* Type to encode        */
```

```
    asn_enc_rval_t ec;                                  /* Encoder return value  */

    /* Allocate the Rectangle_t */
    rectangle = calloc(1, sizeof(Rectangle_t)); /* not malloc! */
    if(!rectangle) {
      perror(''calloc() failed'');
      exit(71); /* better, EX_OSERR */
    }

    /* Initialize the Rectangle members */
    rectangle->height = 42;  /* any random value */
    rectangle->width  = 23;  /* any random value */

    /* BER encode the data if filename is given */
    if(ac < 2) {
      fprintf(stderr, ''Specify filename for BER output\n'');
    } else {
      const char *filename = av[1];
      FILE *fp = fopen(filename, ''wb'');    /* for BER output */

      if(!fp) {
        perror(filename);
        exit(71); /* better, EX_OSERR */
      }

      /* Encode the Rectangle type as BER (DER) */
      ec = der_encode(&asn_DEF_Rectangle,
             rectangle, write_out, fp);
      fclose(fp);
      if(ec.encoded == -1) {
        fprintf(stderr,
          ''Could not encode Rectangle (at %s)\n'',
          ec.failed_type ? ec.failed_type->name : ''unknown'');
        exit(65); /* better, EX_DATAERR */
      } else {
        fprintf(stderr, ''Created %s with BER encoded Rectangle\n'',
          filename);
      }
    }

    return 0; /* Encoding finished successfully */
}
```

○ 다음 Source는 Rectangle.h에 정의된 asn_DEF_Rectangle Type Descriptor를 이용하여, Rectangle_t Type의 구조를 디코딩함

```
#include <stdio.h>
#include <sys/types.h>
#include <Rectangle.h>                  /* Rectangle ASN.1 type  */

int main(int ac, char **av) {
    char buf[1024];                          /* Temporary buffer       */
    Rectangle_t *rectangle = 0;      /* Type to decode */
    asn_dec_rval_t rval;                                  /* Decoder return value  */
    FILE *fp;                                 /* Input file handler     */
    size_t size;                               /* Number of bytes read  */
    char *filename;                          /* Input file name */

    /* Require a single filename argument */
    if(ac != 2) {
      fprintf(stderr, "Usage: %s <file.ber>\n", av[0]);
      exit(64); /* better, EX_USAGE */
    } else {
      filename = av[1];
    }
    /* Open input file as read-only binary */
    fp = fopen(filename, "rb");
    if(!fp) {
      perror(filename);
      exit(66); /* better, EX_NOINPUT */
    }

    /* Read up to the buffer size */
    size = fread(buf, 1, sizeof(buf), fp);
    fclose(fp);
    if(!size) {
      fprintf(stderr, "%s: Empty or broken\n", filename);
      exit(65); /* better, EX_DATAERR */
    }

    /* Decode the input buffer as Rectangle type */
    rval = ber_decode(0, &asn_DEF_Rectangle,
      (void **)&rectangle, buf, size);
    if(rval.code != RC_OK) {
      fprintf(stderr, "%s: Broken Rectangle encoding at byte %ld\n",
        filename, (long)rval.consumed);
      exit(65); /* better, EX_DATAERR */
    }

    return 0; /* Decoding finished successfully */
}
```

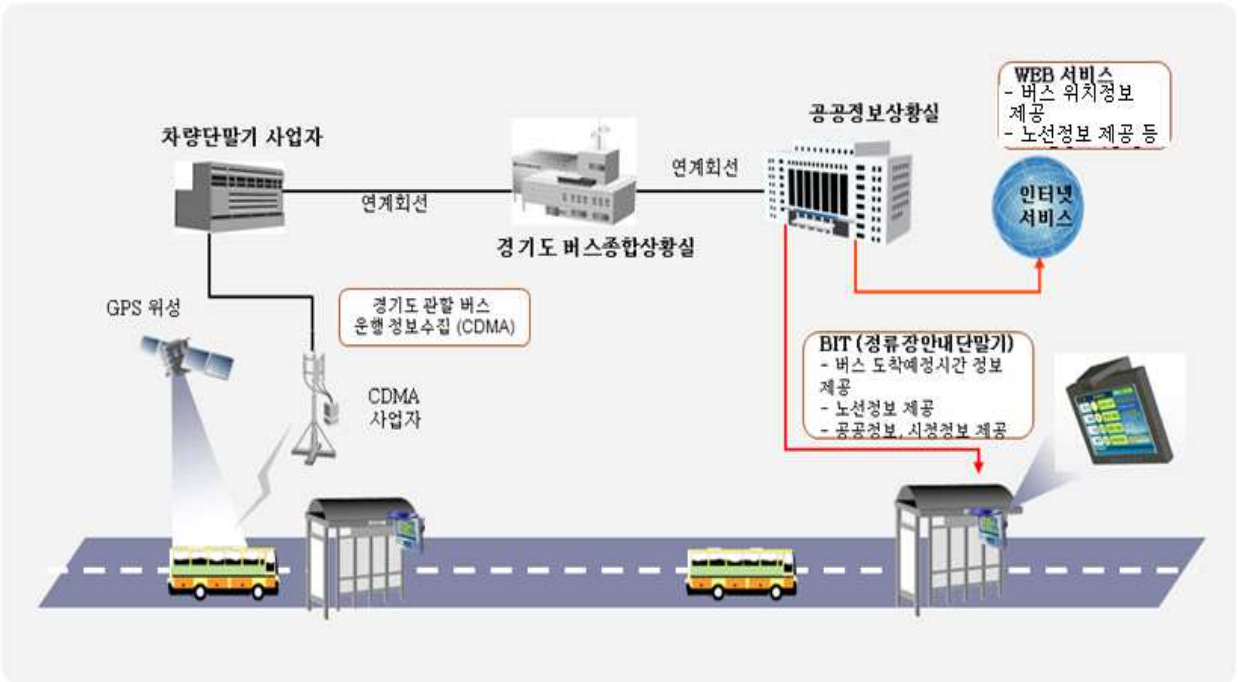
2.4 BIS 개발 시 ASN.1 범용툴 활용 및 사례

2.4.1 애플리케이션 개발 활용

1) BIS 시스템 구성

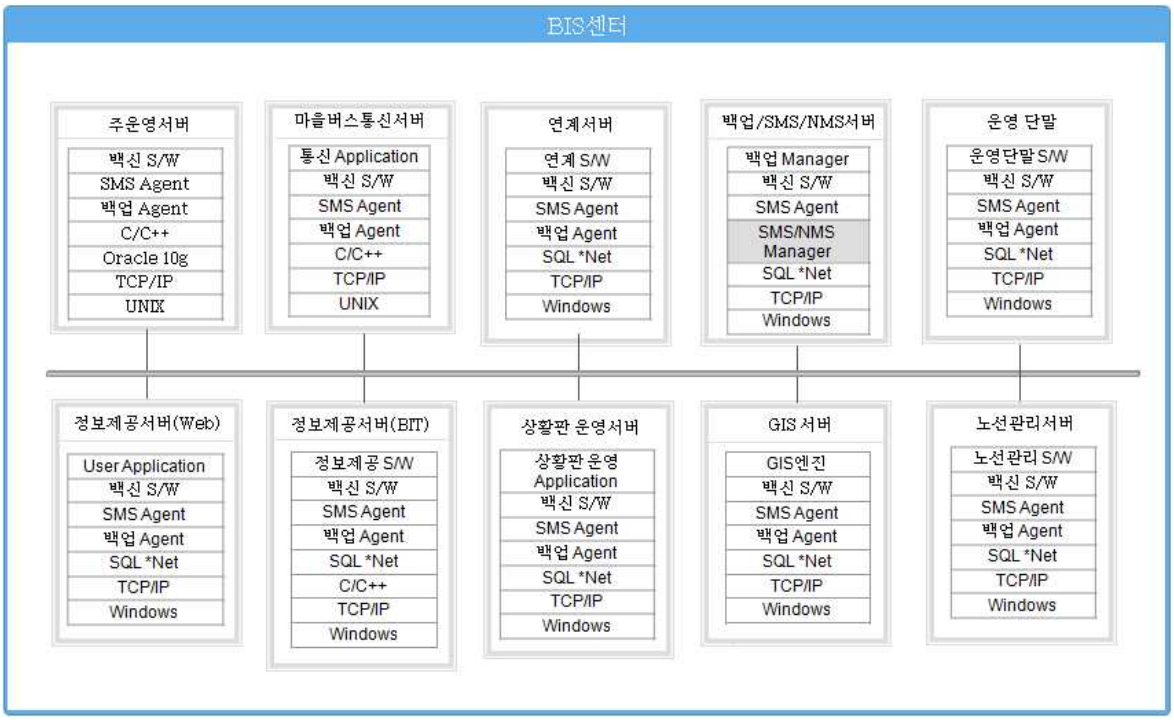
○ BIS 시스템이란 버스의 실시간 위치정보를 수집/가공하여 예측되어진 도착정보를 버스정류소 안내단말기 및 인터넷 정보매체를 통해 실시간으로 이용자에게 제공하는 서비스임

○ 아래 그림은 경기도에 구축된 시스템을 예시로 나타낸 것으로 경기도 버스종합상황실에서 경기도 관내에 있는 모든 시내버스 및 광역버스에 대한 정보(버스정류장 도착·출발, 교차로 통과, 돌발상황정보 등)를 수집하여 각 지자체로 전송(ASN.1)해 주면 각 지차제에서는 이를 가공하여 도착예측정보를 BIT(정류장 안내단말기)로 전송하는 시스템으로 구성되어 있음



2) BIS 센터 소프트웨어

○ BIS 센터 소프트웨어는 주운영서버, 통신서버, 연계서버 등을 포함하여 다음 그림과 같이 구성 됨



- 연계서버 : 경기도 BMS 센터로부터 정보를 수집
- 주운영서버 : 수집된 정보를 가공하여 예측정보를 생성
- 정보제공 서버(BIT, WEB) : 이용자에게 버스 도착정보 및 생활정보를 제공
- 운영단말 : 버스관련 시설물을 관리하고 각종 운영상황을 모니터링
- 상황판 운영서버 : 시스템 및 버스 운영상황을 상황판에 표출하는 시스템
- GIS서버 : Web 및 운영단말에 지도를 관리하는 소프트웨어
- 마을버스 통신서버 : 마을버스 관련정보는 별도 시스템으로 정보를 수집
- 노선관리 서버 : 마을버스 관련 노선을 관리하는 프로그램
- 기타 : SMS서버, ARS서버, OPEN API, 모바일서버

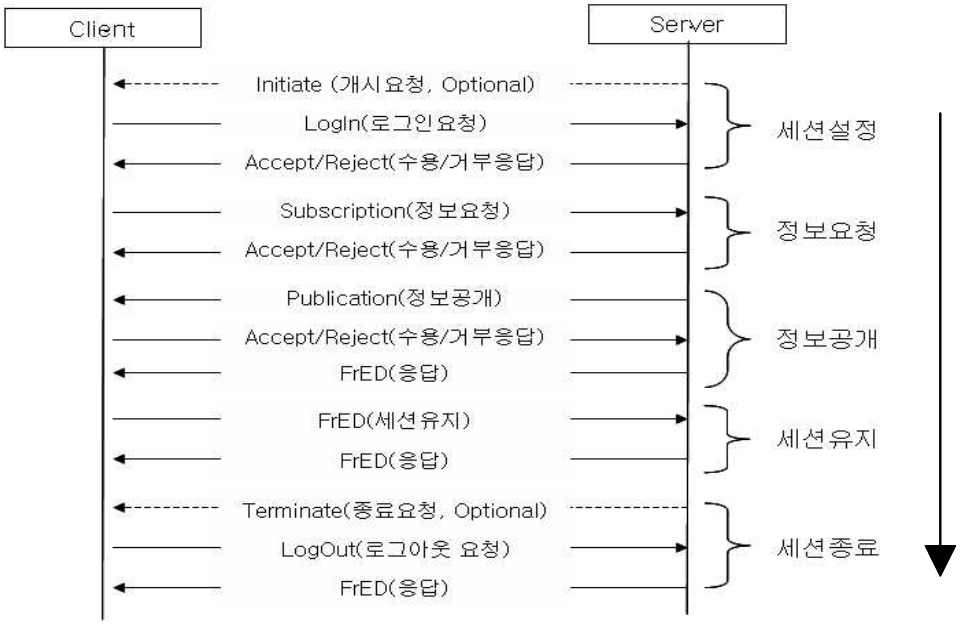
3) ASN.1 적용 기술기준 분석

○ BIS(Bus Information System)는 버스의 실시간 위치정보를 수집/가공하여 예측되어진 도착정보를 버스정류소 안내단말기 및 인터넷 정보매체를 통해 실시간으로 이용자에게 제공하는 서비스임

○ BIS 구축 시 적용해야 하는 기술기준은 '대중교통(버스)정보교환 기술기준'으로, 기술기준에서 규정하고 있는 교환정보와 통신절차를 명확하게 숙지해야 함

○ 또한 '대중교통(버스)정보교환 기술기준'은 정보교환을 위한 통신절차를 다음과 같이 규정하고 있음

- 세션 설정
  - Client에서 로그인 패킷 전송
  - Server에서는 Sender ID, Destin ID, UserName, Password 및 각종 파라미터 적합성 여부를 판단하여 Accept를 전송하거나, Reject인 경우는 Reject 사유를 명시하여 전송
- 세션 유지
  - 세션이 설정된 경우 설정된 Heartbeat duration 시간 안에 Client는 Fred 패킷을 발송하고 서버도 이에 Fred 패킷 전송
  - 일정시간 안에 송수신 패킷 및 Fred 패킷이 없으면 연결 종료
- 세션 종료
  - 세션은 Client 또는 Server에 의해 능동적으로 종료 가능
  - Server인 경우는 종료요청(terminate-request)을, Client인 경우는 종료요청을 받거나 종료를 원할 경우 log-out 패킷을 서버로 전송
  - Log-Out 패킷을 받은 서버는 Fred 패킷을 전송하고 종료
- 정보 요청
  - Client에서 전송이 필요한 패킷에 대하여 한번 요청, 주기적 요청, 이벤트성 요청인지를 명시하여 요청
- 정보 공개
  - Client에서 요청한 정보가 한번 요청, 주기적 요청, 이벤트성 요청인지에 따라 데이터 전송



4) SAMPLE 데이터 패킷 분석

```
30 81 a2 80 01 01 81 81 98 30 81 95 80 01 03 81 01 05 82 01 01 a3 81 84 80 14 32 30
30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 81 14 32 30 30 37 30 38 30 30
00 00 00 00 00 00 00 00 00 00 00 00 82 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00
00 00 00 00 00 00 83 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00
84 14 31 31 30 31 31 30 30 31 00 00 00 00 00 00 00 00 00 00 00 00 00 85 14 30 30 30 30
30 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 a4 03 83 01 01 82 02 d5 9f
```

○ T(Tag) : 0x30

○ L(Length) : 0x81 0xa2

- 데이터 길이는 0~127 범위의 값 사용
- 데이터 길이가 128을 넘어가면 최상위 BIT를 1로 설정하고 그 외 BIT의 뒤에 올 자리수를 설정(ex : 77, 81 a2, 82 01 3d)
- ber_extract_length() 함수는 Value의 길이 값만을 리턴하므로 정확한 패킷의 길이를 알고 싶을 때는 별도의 함수를 만들어 사용

	1st byte	2nd byte	3rd byte	4th byte	5th byte	N
1 byte	'00' to '7F'	-	-	-	-	0 to 127
2 bytes	'81'	'00' to 'FF'	-	-	-	0 to 255
3 bytes	'82'	'0000' to 'FFFF'		-	-	0 to 65 535
4 bytes	'83'	'000000' to 'FFFFFF'			-	0 to 16 777 215
5 bytes	'84'	'00000000' to 'FFFFFFFF'				0 to 4 294 967 295

이미지 출처 : http://www.gorferay.com/ber-tlv-length-fields/

○ V(Value)

- 교통정보교환을 위한 국가표준(KS X ISO 14827-2)에 ASN.1으로 정의된 Datex 데이터 패킷구조 준수

Header부					Data부	Tail부
Dataex Version No.	Authentication Info	Data Packet No.	Data PacketPriority No.	Header Option	PDU	CRC

- Protocol

DatexDataPacket ::= SEQUENCE
{
datex-Version-number ENUMERATED
{
experimental (0),
version1 (1),
…
},
datex-Data OCTET STRING,
datex-Crc-nbr OCTET STRING (SIZE (2))
}
	80 01 01
	-.80 : 데이터테그(T)
	-.01 01 : version1
	(L V)
	
	81 81 98~~~~02 인코딩
	(T L V)
	82 02 d5 9f

C2CAuthenticatedMessage ::= SEQUENCE {
datex-AuthenticationInfo-text OCTET STRING (SIZE (0..255)),
datex-DataPacket-number INTEGER (0..4294967295),
datex-DataPacketPriority-number INTEGER (0..10),
options HeaderOptions,
pdu PDUs
}
	30 81 95
	(T L)
	80 01 03 (TLV) : Fred
	81 01 05 (TLV)
	82 01 01 (TLV)
	a3 81 84 ~~ (TLV)
	a4 03 83 01 01 (TLV)

HeaderOptions ::= SEQUENCE
{
datex-Origin-text UTF8String (SIZE (0..40)) OPTIONAL,
datex-OriginAddress-location OCTET STRING OPTIONAL,
datex-Sender-text UTF8String (SIZE (0..40)) OPTIONAL,
datex-SenderAddress-location OCTET STRING
OPTIONAL,
datex-Destination-text UTF8String (SIZE (0..40)) OPTIONAL,
datex-DestinationAddress-location OCTET STRING OPTIONAL,
datex-Cost Cost OPTIONAL,
datex-DataPacket-time Time OPTIONAL
}	

80 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00 81 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00 82 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00 83 14 32 30 30 37 30 38 30 30 00 00 00 00 00 00 00 00 00 00 00 00 84 14 31 31 30 31 31 30 30 31 00 00 00 00 00 00 00 00 00 00 00 00	datex-Origin-text datex-OriginAddress-location datex-Sender-text datex-SenderAddress-location datex-Destination-text

85 14 30 30 30 30 30 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00	datex-DestinationAddress-location ※ Cost와 Time은 빠짐

PDUs ::= CHOICE
{
datex-Initiate-null [1] Initiate,
login [2] Login,
fred [3] FrED,
terminate [4] Terminate,
logout [5] Logout,
subscription [6] Subscription,
publication [7] Publication,
transfer-done [8] TransferDone,
accept [9] Accept,
reject [10] Reject
}
FrED ::= INTEGER (0..4294967295) -- datexFrED-ConfirmPacket-nbr
	a4 03 83 01 01
	(T L V)
	(T L V)

register unsigned short crc16 = 0;
while(len--) {
crc16 = crctable[((crc16 ^ *buf) & 0xff)] ^ (crc16 >> 8);
buf++;
}
	ISO 3309에 정의된 CRC-16 계산

static unsigned short crctable[ ] = { // CRC-16
0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241, 0xC601, 0x06C0, 0x0780, 0xC741,
0x0500, 0xC5C1, 0xC481, 0x0440, 0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841, 0xD801, 0x18C0, 0x1980, 0xD941,
0x1B00, 0xDBC1, 0xDA81, 0x1A40, 0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641, 0xD201, 0x12C0, 0x1380, 0xD341,
0x1100, 0xD1C1, 0xD081, 0x1040, 0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441, 0x3C00, 0xFCC1, 0xFD81, 0x3D40,
0xFF01, 0x3FC0, 0x3E80, 0xFE41, 0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41, 0xEE01, 0x2EC0, 0x2F80, 0xEF41,
0x2D00, 0xEDC1, 0xEC81, 0x2C40, 0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041, 0xA001, 0x60C0, 0x6180, 0xA141,
0x6300, 0xA3C1, 0xA281, 0x6240, 0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41, 0xAA01, 0x6AC0, 0x6B80, 0xAB41,
0x6900, 0xA9C1, 0xA881, 0x6840, 0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40, 0xB401, 0x74C0, 0x7580, 0xB541,
0x7700, 0xB7C1, 0xB681, 0x7640, 0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241, 0x9601, 0x56C0, 0x5780, 0x9741,
0x5500, 0x95C1, 0x9481, 0x5440, 0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841, 0x8801, 0x48C0, 0x4980, 0x8941,
0x4B00, 0x8BC1, 0x8A81, 0x4A40, 0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641, 0x8201, 0x42C0, 0x4380, 0x8341,
0x4100, 0x81C1, 0x8081, 0x4040
};	

5) 프로그램 샘플

```
g_DatexDataPacket = NULL;
g_AuthenticationMessage = NULL;

dc = ber_decode(NULL, &asn_DEF_DatexDataPacket, (void **)&g_DatexDataPacket, data, size);
```

```
if(dc.code == RC_OK) // Decode Success
{
   //버전정보 추출
   iVersionNbr = g_DatexDataPacket->datex_Version_number;

   memcpy(strCrc,(char *)g_DatexDataPacket->datex_Crc_nbr.buf,g_DatexDataPacket->datex_Crc_nbr.size);
   memcpy(strC2cAuth, (char *)g_DatexDataPacket->datex_Data.buf,g_DatexDataPacket->datex_Data.size);
   iC2cAuthSize = g_DatexDataPacket->datex_Data.size;

   crc16_ccitt(strC2cAuth,iC2cAuthSize, crc);
   if(crc != strCrc) return;

   dc = ber_decode(0, &asn_DEF_C2CAuthenticatedMessage,
                                                (void **)&g_AuthenticationMessage,strC2cAuth,iC2cAuthSize);

   if(dc.code == RC_OK)
   {
         char      strAuthenticationInfo[256] = {0,};
         long      PacketNum, Accept_PacketNum, Reject_PacketNum, packetmask=0;
         int               Priority, psize, msk;

         // C2CAuthenticateMessage 의 datex_AuthenticationInfo_text 데이터 추출
         memcpy(strAuthenticationInfo, (char *)g_AuthenticationMessage->datex_AuthenticationInfo_text.buf,
                                          g_AuthenticationMessage->datex_AuthenticationInfo_text.size);
         // C2CAuthenticateMessage 의 datex_DataPacket_number 데이터 추출
         asn_INTEGER2long(&g_AuthenticationMessage->datex_DataPacket_number, &PacketNum);
         Priority = g_AuthenticationMessage->datex_DataPacketPriority_number;

         // C2CAuthenticateMessage 의 Header 정보 추출
         if(g_AuthenticationMessage->options.datex_Origin_text->size > 0) ;
         if(g_AuthenticationMessage->options.datex_OriginAddress_location->size > 0) ;
         if(g_AuthenticationMessage->options.datex_Sender_text->size > 0) ;
         if(g_AuthenticationMessage->options.datex_SenderAddress_location->size > 0) ;
         if(g_AuthenticationMessage->options.datex_Destination_text->size > 0) ;
         if(g_AuthenticationMessage->options.datex_DestinationAddress_location > 0) ;

         switch(g_AuthenticationMessage->pdu.present)
         {
            case  PDUs_PR_NOTHING: /* No components present */
            break;
            case  PDUs_PR_datex_Initiate_null:
                  break;
            case  PDUs_PR_login: // 서버에 접속하기 위한 클라이언트의 로그인 데이터 패킷
                  g_Login = g_AuthenticationMessage->pdu.choice.login;
                  memcpy(sender, g_Login.datex_Sender_txt.buf, g_Login.datex_Sender_txt.size);
                  memcpy(destin, g_Login.datex_Destination_txt.buf, g_Login.datex_Destination_txt.size);
                  memcpy(user, g_Login.datexLogin_UserName_txt.buf, g_Login.datexLogin_UserName_txt.size);
                  memcpy(passwd, g_Login.datexLogin_Password_txt.buf, g_Login.datexLogin_Password_txt.size);
                  RcvHeartbeatTime = g_Login.datexLogin_HeartbeatDurationMax_qty;
                  HeartbeatCount = RcvHeartbeatTime;
                  RcvResponseTime = g_Login.datexLogin_ResponseTimeOut_qty;
                  Datagram = g_Login.datexLogin_DatagramSize_qty;
                  oid_cnt = g_Login.datexLogin_EncodingRules_id.list.count;

                  break;
            case  PDUs_PR_fred:  // 서버와 크라라이언트의 연결을 유지하기 위한 확인 데이터 패킷
```

```
                    break;
                case  PDUs_PR_terminate:  // 연결을 종료하고자 할때, 서버에서 클라이언트에게 요청하는 데이터 패킷
                    break;
                case  PDUs_PR_logout: //접속을 종료하기 위한 클라이언트의 로그아웃 데이터 패킷
                    break;
                case  PDUs_PR_subscription: /* 클라이언트가 서버에 정보를 요청할 경우 송신하는 데이터 패킷 */
                    break;
                case  PDUs_PR_publication:
                    break;
                case  PDUs_PR_transfer_done:
                    break;
                case  PDUs_PR_accept:
                    break;
                case  PDUs_PR_reject:
                    break;
            }
    }
    else
    {
            printf(" C2CAuthenticatedMessage DeCode Error..\n");
    }
}
else
{
     printf(" DatexDataPacket DeCode Error..\n");
}

// 데이터 구조 FREE
asn_DEF_C2CAuthenticatedMessage.free_struct(&asn_DEF_C2CAuthenticatedMessage,g_AuthenticationMessage,0);
asn_DEF_DatexDataPacket.free_struct(&asn_DEF_DatexDataPacket,g_DatexDataPacket,0);
```

2.4.2 ASN.1 사용시 주의 할 점

1) Length 산출시 주의점

○ ber_extract_length()함수의 리턴값은 Value의 길이만을 리턴한 값으로 Length의 가변적인 길이 값을 추가하여 산출해야 함

2) Long Long형 데이터 입력

○ 현재 Long Long형(4바이트 이상) 데이터를 입력할 수 있는 방법이 없으므로 기존함수를 수정하여 신규 함수를 만들어야 함

```
int AsnControl::asn_long2INTEGER_EDIT(INTEGER_t *st, long long value) {
uint8_t *buf, *bp;
uint8_t *p;
uint8_t *pstart;
```

```
uint8_t *pend1;
int littleEndian = 1;           /* Run-time detection */
int add;

if(!st) {               return -1;          }
buf = (uint8_t *)malloc(sizeof(value));
if(!buf) return -1;

if(*(char *)&littleEndian) {
        pstart = (uint8_t *)&value + sizeof(value) - 1;
        pend1 = (uint8_t *)&value;
        add = -1;
} else {
        pstart = (uint8_t *)&value;
        pend1 = pstart + sizeof(value) - 1;
        add = 1;
}

for(p = pstart; p != pend1; p += add) {
        switch(*p) {
        case 0x00: if((*(p+add) & 0x80) == 0)
                                continue;
                break;
        case 0xff: if((*(p+add) & 0x80))
                                continue;
                break;
        }
        break;
}
for(pstart = p, bp = buf, pend1 += add; p != pend1; p += add)
        *bp++ = *p;
st->buf = buf;
st->size = bp - buf;
return 0;
}
```

3) UTF8 Decoding

○ 한글의 경우 UTF8 타입에서 입력 및 출력하는 경우 별도의 인코딩과 디코딩이 필요함

```
asd = getRouteSchedulePlanning->tpif_SubRouteNameText;
CString RoadWayName = utf8Decoding(asd->buf);
CString AsnControl::utf8Decoding(uint8_t *utf8)
{
	int size = MultiByteToWideChar(CP_UTF8, 0,  (LPCSTR)(utf8), -1, NULL, 0);
	LPWSTR wStr = new WCHAR[size];

	MultiByteToWideChar(CP_UTF8, 0,  (LPCSTR)(utf8), -1, wStr, size);

	CString str = W2CT(wStr);

	delete[] wStr;

	return str;
}
```

4) 패킷 메모리 해제

○ 메모리 해제를 수행하지 않으면 기하급수적으로 메모리 사용량이 늘어나기 때문에 인코딩 또는 디코딩 후에는 항상 메모리 해제를 수행함

```
예)
asn_DEF_C2CAuthenticatedMessage.free_struct(&asn_DEF_C2CAuthenticatedMessage,g_Authentic
ationMessage,0);
asn_DEF_DatexDataPacket.free_struct(&asn_DEF_DatexDataPacket,g_DatexDataPacket,0);
asn_DEF_Message0_MessageBody.free_struct(&asn_DEF_Message0_MessageBody,getMsg0,0);
```