DNS 서버 운영지침서

2006. 8

한국인터넷진흥원

<목 차>

1장 DNS 일반

1.1 DNS 개요 1
1.2 DNS 구조 4
1.3 도메인(Domain)과 존(Zone) 개요 9
1.4 DNS 리소스 레코드 개요 14
1.5 DNS 네임서버 및 리졸버 개요 28
1.6 DNS 도메인 위임설정 개요 31
1.7 도메인 네임 리졸루션(resolution) 37
1.8 관련 표준 목록 40

2장 BIND

2.1 BIND 소개 42
2.2 BIND 설치하기 44
2.3 BIND 설정하기 46
2.4 BIND 운영 61

3장 Windows 2003 DNS 서버

3.1 Windows 2003 DNS 서버 설치 72
3.2 윈도우 2003 DNS 서버 구성 74
3.3 윈도우 2003 DNS 서버 설정 85

4장 Q&A

4.1 BIND 관련 Q&A 89
4.2 네임서버 관련 Q&A 102

1장 DNS 일반

1.1 DNS 개요

도메인 네임 시스템(Domain Name System)은 네트워크 계층이 이해하는 IP 주소와 사람이 이해하고 쉽게 기억할 수 있는 네임(name)을 상호 변환해주는 분산 데이터베이스 시스템으로 인터넷의 모태가 되었던 ARPANET의 발전과정에서 탄생하였다.

ARPANET의 본래 목적은 비싼 컴퓨팅 자원을 공유할 수 있게 하는 것이었는데, 초창기 소수의 호스트들만 연결되었던 환경에선 각 호스트에 대해 숫자 형식의 네트워크 주소를 직접 사용하였다. 그러나 점차 호스트의 수가 늘어나면서 보다 이해하기 쉽고 기억하기 쉬운 호스트 니모닉(mnemonic)의 사용이 도입되었다.

이는 보다 광범위한 범주에서 사용될 수 있는 도메인 네임 체계로 발전하여 현재의 인터넷 네임 체계인 DNS가 등장하는 계기가 된다.

1970년대 초반, TCP/IP 프로토콜이 개발되기 시작하던 시기의 ARPANET에는 23개 정도의 호스트가 연결되어 있었다. IP 프로토콜이 사용되기 전에는 NCP(Network Control Protocol)이라는 호스트-호스트 통신 프로토콜이 사용되고 있었는데 이 시기만 하더라도 도메인 체계에 대한 필요성이 그리 크지 않았다.

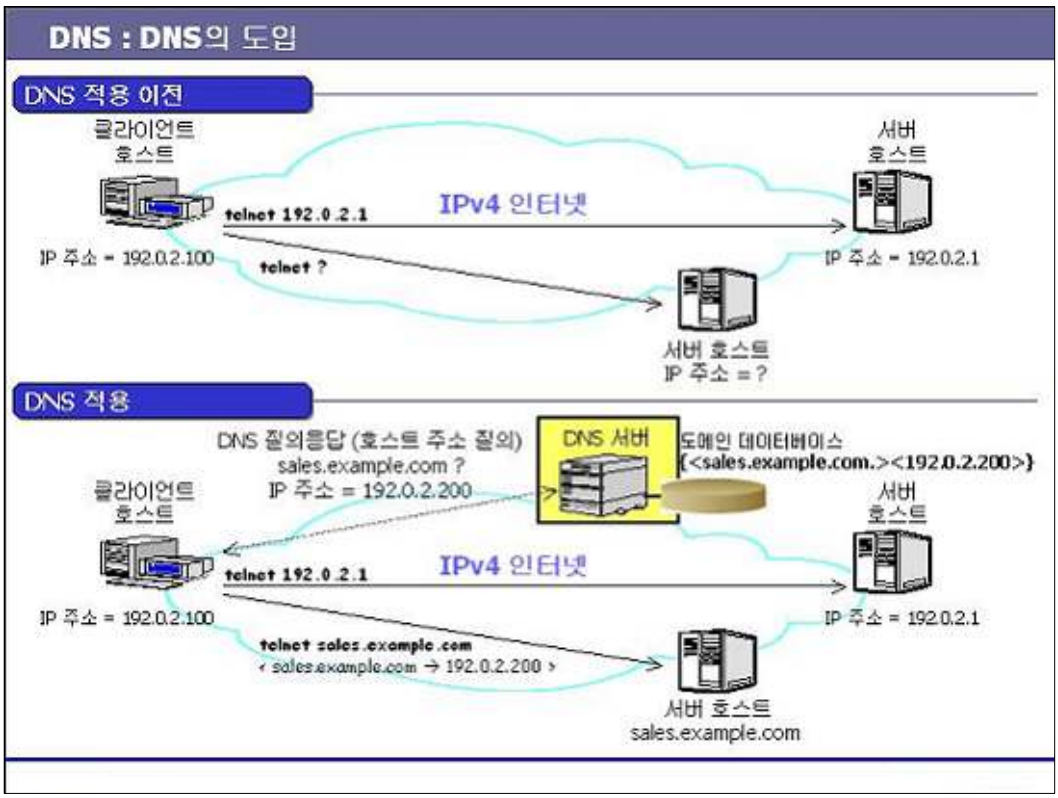
1970년대 중반 이후, TCP/IP가 DoD(Department of Defense)에 적용 시험되고 이어 ARPANET에 적용되었을 때, ARPANET에 연결된 호스트 수는 약 111개 정도로 늘어났다. 이 당시 많은 수의 호스트 네트워크 주소들을 일일이 외울 수가 없으므로 이를 관리하는 방법으로 HOSTS.TXT라는 호스트 니모닉과 네트워크 주소 매핑을 위한 파일을 사용하기 시작했다. 이 HOSTS.TXT는 ARPAnet NIC (Network Information Center)에서 관리하며 각 호스트에서는 NIC이 관리하는 호스트 서버에 FTP로 접속하여 최신 HOSTS.TXT 파일을 다운받아 사용하는 방식이었다.

현재 대다수의 호스트에서 사용하는 hosts 파일은 이러한 [네트워크 주소:호스트 네임] 테이블 관리방식에 상응하는 것으로 호스트의 네임 변환체계 속에 포함되어 사용되고 있다.

1980년대 초반, ARPANET에 연결된 호스트 수는 약 562개였고 HOSTS.TXT 파일에 대한 갱신작업은 이제 한계에 이르렀다. 기존의 HOSTS.TXT 파일을 통한 호스트 네임 관리체계를 넘어 보다 체계적이고 효율적인 네임체계 및 관리 방안이 필요하게 되었다. 그리하여 1983년 위스콘신 대학에서 DNS 체계를 만들었다.

DNS는 도메인 네임체계와 이를 구현하는 계층적인 분산 도메인 데이터베이스 체계로 개발되었고 도메인 네임체계를 구성하는 도메인 네임은 하나의 데이터베이스에서 관리하기보다는 계층적이고 분산구조를 갖는 데이터베이스 체계로 고안되었다. 계층적인 네임체계는 추가 확장이 용이한 구조를 제공하고 분산구조의 데이터베이스는 네임 데이터베이스에 대한 분산 관리체계를 가능케 함으로써 그 관리의 효율성을 증대시켰다.

그럼, 여기서 DNS 도입으로 무엇이 얼마나 달라졌는지 알아보도록 하자. 그림 1-1은 DNS 도입 적용전과 적용후의 모습을 보여준다.



[그림 1-1] DNS의 도입

DNS가 적용되지 않은 환경에서는 인터넷의 네트워크 주소를 직접 사용하여야 원하는 호스트와 통신할 수 있다. 그러나 인터넷 상에는 수많은 호스트가 존재한다.

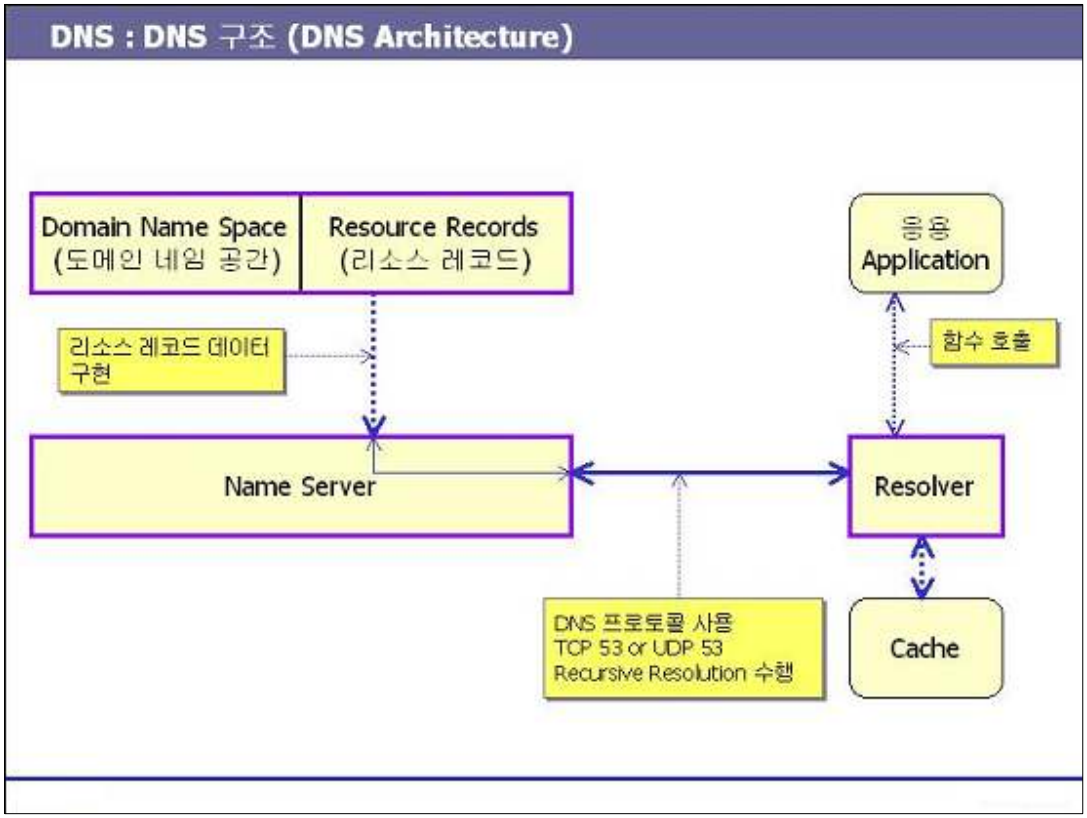
특정 호스트가 어떤 IP 주소를 가지고 있는지, 어떤 서비스를 제공하는지를 사전에 알지 못한 상태에서는 해당 호스트로 접근하는 방법을 파악할 수 없다.

DNS가 적용되면 상황은 달라질 수 있다. DNS는 계층적인 네임체계를 제공하므로 미국의 버클리 대학의 웹 사이트에 대한 접근을 원하는 경우, 버클리 대학의 영문 표기인 Berkeley와 교육 기관용 도메인인 edu를 조합하여 Berkeley.edu를 도메인 네임으로 사용, 접속을 시도함으로써 버클리대학의 호스트로 접근하는 것이 가능할 수 있다.

이때, 인터넷 상에는 그림처럼 인터넷 도메인 네임 데이터베이스를 구성하는 DNS 서버가 체계적으로 구축, 존재해야만 한다. 각 클라이언트 호스트는 사용자가 지정한 도메인 네임의 호스트에 접속을 개시하기 이전에 먼저 도메인 네임을 네트워크 주소인 IP 주소로 변환하는 절차를 DNS 서버를 사용하여 수행한다. IP 주소를 파악한 이후에는 해당 IP 주소를 사용하여 일반적인 네트워크 통신에 필요한 절차를 개시한다.

1.2 DNS 구조

DNS는 크게 도메인 네임 공간(Domain Name Space) 및 리소스 레코드(Resource Record), 네임서버(Name Server), 리졸버(Resolver)의 3가지 기능 요소로 구성된다.



[그림 1-2] DNS의 구조

도메인 네임 공간(Domain Name Space)은 흔히 인터넷에서 사용되고 있는 도메인 네임의 계층적 구조 공간을 의미한다. www.example.com은 이 도메인 네임 공간에 속한 하나의 노드 즉, 호스트 네임을 지칭한다. 도메인 네임 공간의 분배와 도메인 네임의 할당은 전 세계적으로 도메인 네임 할당기관을 통해 체계적으로 할당, 관리되고 있다. 이를 통해 인터넷 상에서 특정 도메인 네임은 언제나 유일한 네임(globally unique name)으로써 유지된다. 도메인 네임 공간은 네임 주소 영역을 분배, 할당, 구성하는 방식을 제공한다. 그리고 이러한 도메인 네임 공간은 트리(tree)형태의 계층적인 구조를 이루어져있다.

리소스 레코드(resource record)는 도메인 네임 공간 중에서 지정된 도메인

네임에 대해 필요한 인터넷 자원(resource) 정보를 매핑하는 수단을 제공한다. 예를 들어 www.example.com의 IP 주소가 192.0.2.101일 경우, 이를 인터넷 상에 알려주기 위해서는 www.example.com이라는 네임(name)에 대해 IP 주소 속성을 연결시켜 DNS 데이터베이스를 구성하여야 한다. 이는 zone 파일을 사용하여 www.example.com 1800 IN A 192.0.2.101 이라는 형태의 표기를 사용하여 설정한다. 이러한 하나의 도메인 네임(domain name)이 가지는 속성 정보를 지정하는 수단으로 정의된 것이 리소스 레코드이다.

리소스 레코드는 확장이 가능하다. 즉, IPv4 네트워크 주소정보를 설정하기 위해 A type의 리소스 레코드가 사용되고 있으나 이제 IPv6가 도입됨에 따라 IPv6 네트워크 주소정보를 설정하기 위해 AAAA type의 리소스 레코드가 추가로 정의되었다. 리소스 레코드의 확장은 기존에 정의된 리소스 레코드에 대한 영향 없이 이루어진다.
리소스 레코드는 도메인 네임 시스템 체계에 있어 도메인 데이터베이스를 구성하는 역할을 한다.

네임서버는 도메인 존(domain zone)의 정보를 소유하고 이에 대한 질의에 대해 응답하는 역할을 수행한다. 흔히 DNS 서버라고도 하며 특히 특정 질의에 대해 자신이 소유한 도메인 존(domain zone)의 정보만 그 응답으로 제공하는 동작을 하는 DNS 서버라는 의미로써 iterative DNS 서버라고 불리기도 한다. 또한 도메인 네임 공간상의 특정 영역(zone)에 대해 관리권한이 위임된 서버이므로 authoritative DNS 서버, authoritative 네임서버라고 한다.
네임서버는 인터넷 상의 도메인 네임에 대해 분산된 도메인 데이터베이스를 구성한다. 이때 도메인 데이터베이스는 각 네임서버에 다양한 형태의 분산구조로 설정된 리소스 레코드들로 이루어진다.

리졸버(Resolver)는 네임서버에 의해 구성된 도메인 데이터베이스를 검색하는 역할을 한다. 리졸버는 응용 애플리케이션 프로그램과 도메인 네임 시스템간의 인터페이스 역할을 담당한다. 또한 요청된 리소스 레코드가 존재하는 위치를 도메인 네임 시스템에서 찾고 이 리소스 레코드의 정보를 최종 응답으로 되돌려 주는 기능을 담당한다.
리졸버는 전체 도메인 네임 시스템에 대해 전체 도메인 데이터베이스를 검색할 수 있도록 도메인 네임 시스템 검색의 시작점이 되는 루트 네임서버(root name server)의 IP 주소 정보를 자신의 환경 구성 파일로 가지고 있다.

또한 리졸버는 동일한 DNS 질의를 짧은 시간 내에 빈번하게 반복하는 것을

방지하기 위해 캐시(cache)를 내부에 구현하여 DNS 질의 결과 데이터를 일정 기간 동안 이 캐시에 저장하여 관리한다. 각 리소스 레코드는 레코드 자체에 지정된 TTL(Time To Live) 시간 동안만 리졸버의 캐시에 존재한 후 삭제된다. 그러나 이러한 리졸버의 전체 기능을 모든 호스트에 구현해야 하는 것은 아니다. 현재 대부분의 호스트에는 전체 리졸버 기능이 아닌 스터브 리졸버 형태로 구현하고 있다.

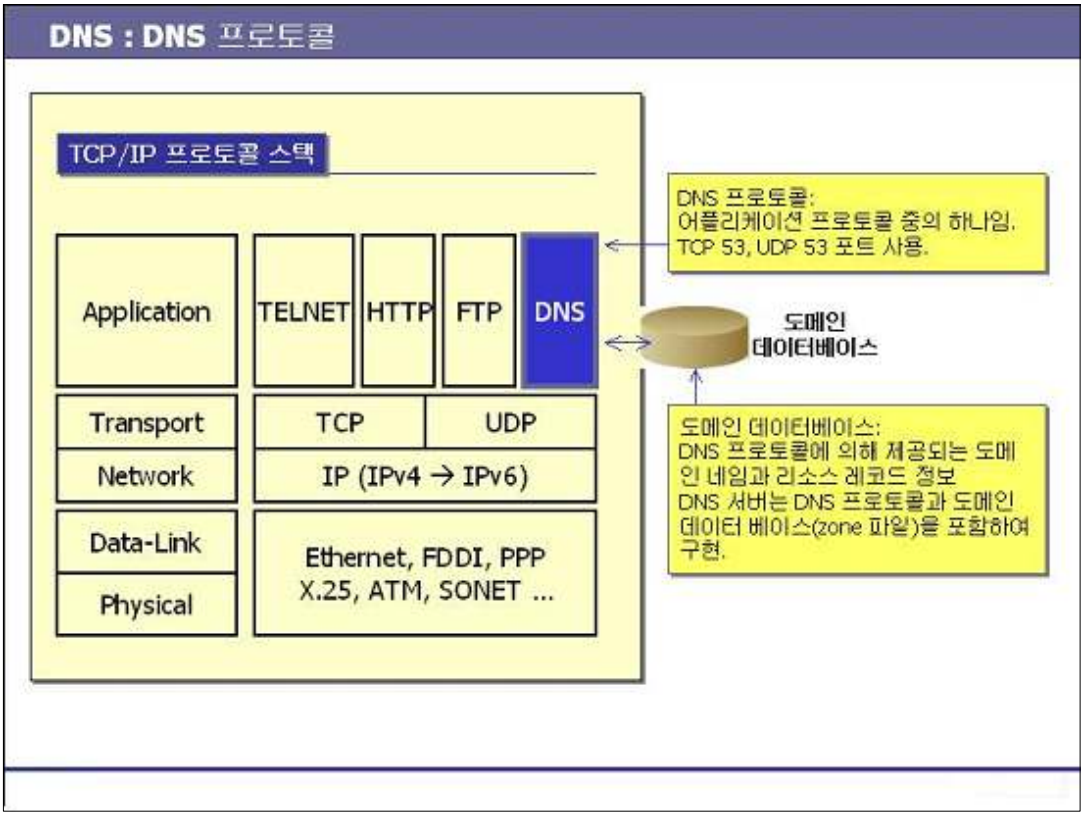
스터브 리졸버(Stub Resolver)는 호스트 내의 응용 프로그램에 프로그래밍 인터페이스를 제공하고 응용 프로그램의 네임 질의 요청이 있는 경우, 시스템에 지정된 리졸버 역할의 네임서버로 DNS 질의를 요청한다. 리졸버는 스터브 리졸버를 대신하여 전체 도메인 네임 시스템에 대하여 요청된 레코드가 존재하는 네임서버를 파악하고 해당 리소스 레코드 정보를 획득한 후 최종 응답결과를 스터브 리졸버에게 전달한다. 여기에서 스터브 리졸버는 전체 도메인 네임 시스템(DNS)에 대해 원하는 리소스 레코드의 위치를 파악할 수 있는 알고리즘을 구현하지 않는다. 대신 항상 DNS 요청을 처리할 수 있는 리졸버 DNS 서버(resolver DNS server)의 IP 주소를 지정하는 것으로 충분하다. 흔히 PC에 설정하는 ISP 사업자가 제공하는 DNS 서버 또는 네임서버는 이 리졸버 기능이 설정된 서버를 지칭한다. UNIX 계열 호스트의 경우에는 /etc/resolv.conf 파일이 리졸버 서버(resolver server)의 IP 주소를 설정하는 환경 구성파일에 해당한다. Windows 계열 호스트는 '내 네트워크 환경'의 등록정보에서 인터넷 프로토콜 등록정보에서 IP 주소를 설정하면 된다(구체적인 설정법은 Q&A를 참조한다).

일반적으로 널리 쓰이고 있는 BIND는 네임서버와 리졸버를 구현한 DNS 서버 소프트웨어이다. 도메인 데이터베이스를 구성하는 리소스 레코드는 존(zone) 파일로 지정하며 여기에 작성된 텍스트 형태의 리소스 레코드 데이터는 DNS 데몬(daemon)의 구동시 그 텍스트 표기 내용을 해석하여 데이터베이스 형식으로 변환, 메모리에 리소스 레코드 데이터베이스를 구성한다.
BIND의 경우, 네임서버와 리졸버를 모두 네임 데몬(name daemon)에 포함하고 있다. 따라서 BIND는 용도에 따라 네임서버로만 동작하게 하거나, 네임서버와 리졸버 모두의 기능을 수행하도록 설정, 운영할 수 있다.

이상과 같이 도메인 네임 시스템(DNS)은 도메인 네임체계와 각 도메인 네임에 대한 속성 정보인 리소스 레코드들을 네임서버 소프트웨어로 구현하여 시스템화 하였다. 또한 분산 구조의 도메인 네임 데이터베이스에 접근하여 원하는 도메인

네임의 리소스 레코드 데이터를 검색하기 위한 수단으로써 소프트웨어로 구현된 리졸버를 정의하였다. 그리고 최종 단말인 호스트에서는 도메인 네임에 대한 주소 변환을 위해 호스트 내부에 구현된 스터브 리졸버를 사용하여 원하는 네임 변환 요청을 함으로써 전체 인터넷 상의 도메인 데이터베이스에서 원하는 정보를 얻는다.

DNS의 구성요소 간에는 상호 도메인 데이터베이스의 검색과 응답을 위한 프로토콜이 필요하다. DNS 프로토콜은 애플리케이션 계층에 속하는 애플리케이션 프로토콜이다.



[그림 1-3] DNS 프로토콜

DNS 프로토콜은 네트워크를 경유하여 DNS 구현요소 간에 DNS 질의(DNS query)와 응답(DNS response)을 수행하기 위한 서버-클라이언트 모델의 애플리케이션 프로토콜이다. DNS 프로토콜은 그림과 같이 {TCP,UDP}/IP 프로토콜 스택 위에 존재한다.

DNS 프로토콜은 TCP 및 UDP 포트 번호 53번을 사용한다. 네임서버는 TCP와

UDP 53번 포트를 디폴트 포트(default port)로 하여 네트워크 상의 요청에 대기한다.
DNS 질의의 대부분은 UDP 포트 53을 사용하여 질의와 응답이 이루어진다. 그러나 UDP 헤더 이후의 DNS 헤더를 포함한 DNS 메시지 영역의 길이가 512 바이트를 초과하는 경우, TCP 53번 포트를 사용하는 TCP 연결을 통한 DNS 질의응답이 이루어지는 메커니즘이 존재한다. 이외에 TCP가 사용되는 경우는 동일한 도메인 존(domain zone)을 가지고 있는 네임서버간의 도메인 존(domain zone) 데이터 송수신을 위한 존 트랜스퍼(zone transfer)를 수행하는 경우이다. 이 경우에는 많은 데이터의 전송이 필요하므로 TCP를 사용한 연결을 통해 데이터 전송요청이 이루어진다.
DNS 프로토콜은 네임서버와 리졸버 간, 그리고 리졸버와 스터브 리졸버 간에 사용된다. 스터브 리졸버와 응용 프로그램은 프로그래밍 인터페이스를 사용하며 DNS 프로토콜을 사용치 않는다.

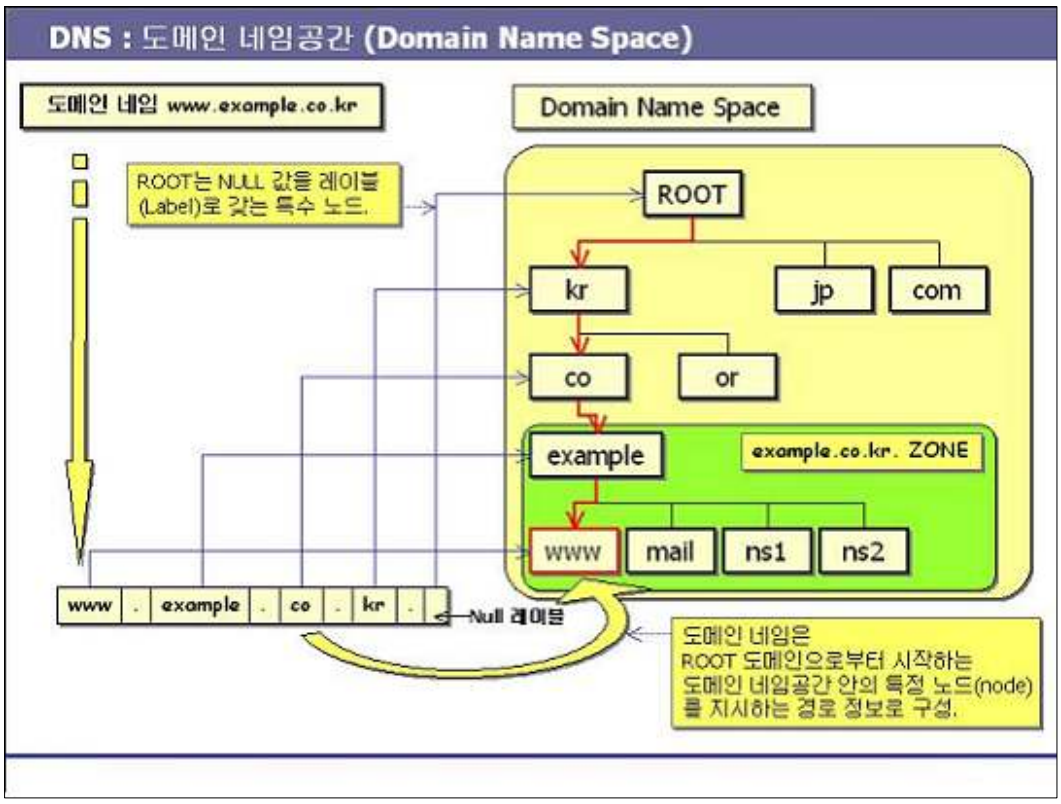
1.3 도메인(Domain)과 존(Zone) 개요

DNS는 도메인 네임체계를 기반으로 한다.

DNS의 구성요소 중 도메인 네임 공간(Domain Name Space)은 인터넷 상에 존재하는 리소스(resource)에 대한 네임체계를 어떻게 체계화할 것인가에 대한 구성방안이라 할 수 있다.

DNS의 계층적(hierarchical)인 도메인 네임 구조는 루트 도메인(root domain)으로부터 시작하여 각 서브 도메인으로 위임(delegation)하는 구조로 형성한다.

최상위의 루트 도메인은 특수한 도메인으로써 모든 도메인의 부모 도메인(parent domain)이다. 모든 도메인은 루트 도메인으로부터 위임받은 서브 도메인(sub domain)으로 정의된다.



[그림 1-4] DNS 도메인 네임 공간

도메인 네임은 “.”(dot)로 구분된 레이블(label)로 구조화된 이름으로 표시한다.

이 이름에는 루트 도메인으로부터 해당 노드에 이르는 경로상의 각 계층(level)을 모두 표시함으로써 도메인 네임 공간상에서의 그 위치를 표시한다. 예를 들어 도메인 네임 www.example.co.kr은 kr 도메인에서 co 도메인으로 위임된 후 다시 위임된 example 도메인에 속하는 www 라는 네임(name)을 지닌 노드 또는 호스트를 의미한다. 이로써 도메인 네임은 그 자체로서 도메인 네임 공간에서 유일한 이름을 지닌다.

이 예에서 www, example, co, kr은 계층적 구조의 각 수준(level)에 해당하는 레이블이다. 레이블들은 점(dot)으로 구분되어 있다. 최상위 도메인인 루트 도메인에 해당하는 레이블은 널(null) 레이블로서 그 길이가 0인 특수한 레이블로 정의하고 있다. 널(null) 레이블은 루트 도메인만이 갖는 예약된 레이블로서 타 도메인은 이를 레이블로 사용할 수 없다.

레이블은 0~63 옥텟(octet)의 크기를 가지는 숫자/문자의 조합으로 부여하도록 정하고 있다. 그러나 이중에서 0의 크기를 가지는 널(null) 레이블은 루트 노드만이 사용하는 레이블로 다른 노드에는 이 널(null) 레이블을 사용할 수 없다. 따라서 루트 노드가 아닌 다른 노드들에는 1~63 옥텟(octet) 크기의 레이블을 부여할 수 있다.

또한 레이블에 사용할 수 있는 문자 숫자 기호 코드에도 제한이 정해져 있다. 레이블에는 ASCII 코드 중 “a”에서 “z”까지(또는 “A”에서 “Z”까지)의 문자, 그리고 “0”에서 “9”까지의 숫자, 그리고 기호문자 중 “-”(hyphen)만을 사용할 수 있다. 이외의 ASCII 코드는 레이블에 사용할 수 없다. 따라서 “kim&park.co.kr”이나 “make$.com”, “save_money.co.kr” 등과 같은 도메인 네임은 존재할 수 없다. 또한 레이블의 첫 문자에는 “-”(hyphen)이 사용될 수 없다. “--notice--.co.kr”과 같은 도메인은 생성할 수 없다.

FQDN(Fully Qualified Domain Name)은 도메인 네임을 루트 도메인으로부터 시작하는 전체 이름의 표기를 사용한 것으로 위의 예는 www.example.co.kr.과 같이 도메인 네임의 끝에 루트 도메인의 널(null) 레이블까지 완전히 표기하는 것을 지칭한다. 즉 마지막의 루트 도메인의 레이블이 널(null) 값이므로 “.”을 끝에 표기함으로써 루트 도메인까지의 경로 레이블을 모두 포함하여 도메인 네임에 표기한 것이다.
FQDN의 총길이는 구분자 역할을 하는 점(dot)을 포함하여 최대 255 옥텟(octet)의 길이를 가질 수 있다.

원칙적으로 DNS에서는 도메인 네임은 FQDN을 사용한다. 단지 사용자의 편의를 고려하여 루트 도메인의 표기를 생략한 도메인 네임의 표기를 허용한다. 그러나 시스템 내부에서 도메인 네임에 대한 DNS 질의에 있어 DNS 프로토콜에 사용되는 실제 도메인 네임은 항상 FQDN 형식의 도메인 네임을 사용한다. 만일 호스트 시스템의 기본 도메인이 “company.com.”으로 설정된 경우, 사용자가 입력한 도메인 네임 “www.example.co.kr”은 “www.example.co.kr.company.com.”으로 해석될 수 있다. 이 호스트에 존재하는 시스템의 리졸버 루틴은 사용자가 입력한 “www.example.co.kr”을 “www.example.co.kr.”로 해석하여 DNS 질의절차를 수행하고 만약 이 도메인 네임이 존재하지 않는다는 결과를 얻는 경우에는 이를 다시 “www.example.co.kr.company.com.”으로 해석하여 DNS 질의절차를 수행하는 방법으로 해당 도메인 네임을 해석하려 시도할 수 있다. 곧, FQDN이 아닌 도메인 네임은 상위 도메인 네임이 생략된 도메인 네임으로 해석한다. 이를 PQDN(Partially Qualified Domain Name)이라고 하기도 한다.
그러나 FQDN의 경우에는 전체 도메인이 모두 표기된 도메인 네임으로 해석한다. 이러한 차이는 특히 네임서버의 존(zone) 파일을 작성할 때 주의해야 할 사항으로서 의도하지 않은 문제를 야기할 수 있는 요인으로 작용할 수 있다.

도메인 트리(domain tree)에서 도메인 네임은 도메인 네임 공간에 속한 각 노드(node)의 경로정보를 포함한 이름이다. 노드는 도메인 트리에서 분기점이나 트리의 종단점을 이루고 있다. 각 노드는 하나의 레이블을 갖는다. 트리(tree) 구조로 연결된 노드들은 단일한 노드로부터 연결된 구조를 갖는데, 이 모든 노드의 단일한 뿌리가 되는 노드가 루트 노드(root node)이다. 루트 노드의 레이블은 특수한 레이블 값인 널(null) 레이블을 갖는다. 곧, 루트 노드는 그 이름이 없는 노드라고 할 수 있다.

루트 노드의 하위 노드로서 com, org, net, kr, jp, int 등의 노드가 존재한다.
이러한 노드들을 지칭하여 최상위 도메인(TLD, Top Level Domain)이라 한다. 그리고 이 최상위 도메인 바로 아래 레벨(level)의 노드들을 개념상으로 2단계 도메인(SLD, Second Level Domain)이라 한다. 여기에는 co.kr, or.kr, ne.kr 도메인들이 있다.
최상위 도메인(TLD)은 “일반도메인(generic domain)”, “국가도메인(country code domain)”, “인터넷 인프라 도메인(infrastructure domain)” 으로 구분할 수 있다. 최상위 도메인 중 일반도메인(generic domain)은 gTLD(generic TLD) 라 하고, 국가도메인은 ccTLD(country code TLD) 라 한다. 이외에 인터넷 인프라

도메인은 특수한 도메인으로써 ARPA 도메인을 지칭한다.

아래는 각 범주별로 구분된 최상위 도메인(TLD) 현황을 정리한 것이다.

구분	최상위 도메인(TLD)	비고
gTLD	.aero, .biz, .com, .coop, .info, .museum, .name, .net, .org, .pro, .gov, .edu, .mil, .int	3자리 또는 그 이상 자리수의 레이블을 갖는 범용 최상위 도메인. 각 도메인 별로 그 사용목적을 지니고 있음.
ccTLD	.kr, .jp, .cn, .us 등과 같이 2자리 국가 코드 레이블을 갖는 국가별 도메인	각 국가별 관리 정책에 의해 관리하는 국가별 도메인. 전체 ccTLD 리스트: http://www.iana.org/cctld/cctld-whois.htm
특수 도메인	.arpa	인터넷 기반 구조에 대한 정보를 제공하기 위한 목적의 특수 도메인. e164.arpa, in-addr.arpa, ip6.arpa, uri.arpa, urn.arpa의 하위 도메인 존재

[표 1-1] 최상위 도메인 현황

DNS는 분산구조의 네임체계이다. 이는 분산된 구조의 도메인 데이터베이스를 형성하며 도메인 데이터베이스도 분산 관리 체계로 운영하고 있다. 이러한 분산된 도메인 데이터베이스의 구성과 분산 관리체계를 위해 위임(delegation)이 적용된다.
루트 도메인은 하위의 최상위 도메인(TLD)으로 해당 도메인의 관리를 위임한다. 그리고 최상위 도메인은 다시 2단계 도메인(SLD)을 생성하여 그 관리를 위임한다. .kr 도메인의 경우에는 루트 도메인으로부터 국가별 도메인인 .kr 도메인의 위임을 받고 그 하위에 co.kr, or.kr, go.kr, ne.kr 등의 도메인을 생성하여 도메인 위임을 하고 co.kr 도메인은 다시 국내 기업기관을 대상으로 필요한 도메인의 등록과 그 위임을 하는 구조를 형성하고 있다.

이러한 위임에 있어 필요한 개념이 존(zone)이다. 이는 상위 도메인으로부터 위임받은 도메인을 기준으로 하위 노드를 생성, 관리하고 다시 서브 도메인을 생성 위임 및 관리할 수 있는 도메인 영역(zone)을 의미한다. 각 도메인의 관리자는 해당 도메인의 하위 영역을 독자적으로 관리하는 권한을 지닌다. 즉 위임은 도메인 네임의 생성, 유지, 삭제 등을 할 수 있는 일정한 영역의 도메인에 대한 관리 권한을 위임하는 것을 의미한다.

도메인 네임 시스템(DNS)은 이러한 위임체계를 통해 전 세계 각 사이트에서 위임받은 도메인 영역에 대한 분산된 관리체계를 유지하고 있다. 도메인 등록시, 이 도메인에 대한 일정 기간 동안의 사용권과 함께 해당 도메인 하위의 모든

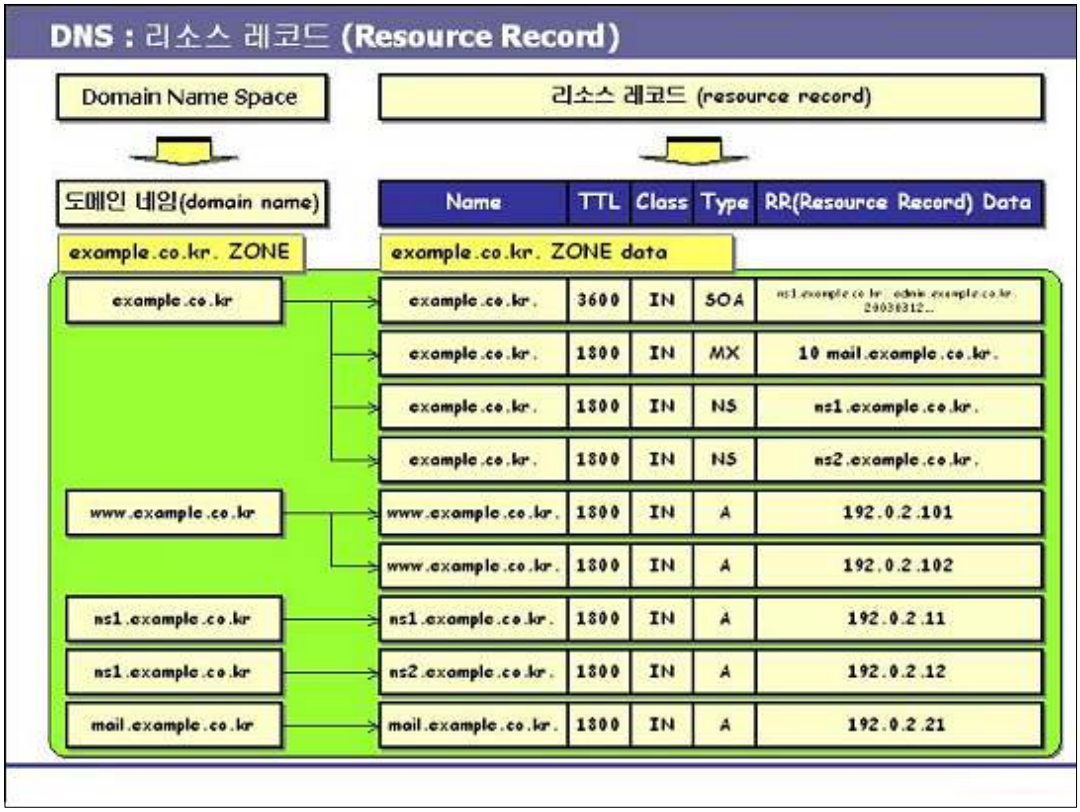
도메인 네임의 생성, 유지, 삭제 등에 대한 전적인 관리 권한을 위임받게 된다. 도메인 존(domain zone)은 도메인 트리 구조에서 특정 노드와 그 하위 노드를 포함하는 일정한 영역을 지칭한다.

“example.co.kr”의 경우, 한국인터넷진흥원(NIDA)로부터 “example.co.kr” 이라는 도메인 등록절차를 마침으로써 해당 도메인에 대한 위임을 받는다. “example.co.kr” 도메인의 위임을 받은 사이트 관리자는 해당 도메인에 속한 새로운 노드를 필요에 따라 원하는 레이블을 부여하여 도메인 네임을 생성할 수 있다. 웹 서버에 대해서는 “www”, 메일 서버에 대해서는 “mail”, 도메인의 DNS 서버에 대해서는 “ns1”과 “ns2"의 레이블을 갖는 노드를 생성, 설정한다. 이 설정작업을 통해 인터넷에는 새로운 도메인 네임 "www.example.co.kr"과 "mail.example.co.kr", "ns1.example.co.kr", "ns2.example.co.kr"이 알려지게 되고 이 도메인 네임을 사용한 접속이 가능하게 된다. 여기에서 "example.co.kr" 은 하나의 도메인 존(zone)이 된다. 이 존(zone)은 단일 관리자에 의해 관리되는 도메인 네임 영역이다.

co.kr은 한국인터넷진흥원(NIDA)가 관리하는 도메인 존(zone)이다. co.kr 도메인은 example.co.kr 도메인을 포함하는 도메인이다. 그러나 co.kr 도메인 존(zone)은 example.co.kr에 대한 위임설정 관리만을 수행하며 example.co.kr 도메인이 가지고 있는 도메인 네임에 대한 관리는 하지 않는다.

1.4 DNS 리소스 레코드 개요

도메인 네임 공간(Domain Name Space)에서 설정된 도메인 네임은 그 자체로서는 인터넷 자원과 관련된 아무런 정보를 가지고 있지 않다. 즉, 하나의 도메인 네임은 전체 네임체계 속에서 유일성을 갖는 네임으로서의 특성만을 가지고 있다.
도메인 네임은 인터넷 상에서 사용하는 각종 자원(resource) 정보와 연관시킬 때, 인터넷 네임체계로서의 의미를 지닐 수 있다.
리소스 레코드(RR : Resource Record)는 도메인 네임(domain name)과 인터넷 자원(resource) 정보를 매핑하여 하나의 분산 데이터베이스를 구성하기 위한 수단이다.



[그림 1-5] DNS 리소스 레코드

리소스 레코드(RR)는 도메인 네임이 갖는 속성을 표현한다. 하나의 도메인 네임은 호스트 네트워크 주소 리소스 레코드(Host Address RR)인 A 타입 RR에 의해 도메인 네임이 지칭하는 IP 주소를 지정할 수 있다.
하나의 도메인 네임은 다수의 리소스 레코드(RR)를 그 속성정보로 가질 수 있다. 예를 들면, www.example.co.kr은 192.0.2.101 및 192.0.2.102의 2개의 IP 주소를 그 속성정보로 가지면서 동시에 "example web site"라는 텍스트 문자열을 텍스트

리소스 레코드(TXT type RR)를 사용하여 그 속성정보로 지닐 수 있다. 이와 같이 하나의 도메인 네임은 인터넷 네트워크 주소를 포함하는 다양한 인터넷 자원(resource)으로서의 속성정보를 설정하여 인터넷 통신에서 유용하게 사용될 수 있다.

인터넷 네트워크 주소는 인터넷 자원(resource) 정보 중의 하나로써 가장 널리 쓰이는 도메인 네임의 인터넷 자원 속성정보이다. IPv6가 도입됨으로써 이제는 하나의 도메인 네임에 의해 지정되는 호스트가 IPv4 주소와 IPv6 주소를 동시에 가지고 있다면 이 도메인 네임에는 A 타입의 리소스 레코드(A type RR)와 AAAA 타입의 리소스 레코드(AAAA type RR)를 함께 설정하게 된다.
인터넷 자원(resource) 정보는 IP 주소를 제외하고도 다양한 정보가 존재한다. 리소스 레코드(RR)는 이러한 다양한 인터넷 자원 정보를 표시하는 체계로서 추가 확장 정의될 수 있는 유연한 구조를 지니고 있다. 앞으로도 다양한 정보가 도메인 네임에 매핑될 수 있도록 추가 정의된 리소스 레코드(RR)가 등장할 것이다.

이와 같이 동일한 도메인 네임에 대해 정의된 다수의 리소스 레코드(RR)를 RRset 이라 한다.

도메인 네임 중에서 IP 주소 속성정보를 갖는 도메인 네임을 특별히 호스트 네임(host name)이라 한다. IP 주소정보를 가지지 않는 도메인 네임이 있을 수 있다. 예를 들어 about.example.co.kr이라는 도메인 네임을 생성하고 여기에 사이트에 안내문을 텍스트 문자열 리소스 레코드(TXT type RR)를 사용하여 사이트의 주소와 안내사항을 포함한 문자열을 정의하여 서비스에 이용할 수 있다. 이 경우, about.example.co.kr은 호스트 네임(host name)이라 하지 않으며 단지 example.co.kr 사이트에 대한 정보를 제공하는 역할을 한다.

리소스 레코드는 도메인 네임에 대해 관련 속성정보를 매핑할 수 있도록 구조화 되어있다. 일반적인 구조는 그림과 같이 **<Name><TTL><Class><Type><RDATA>**로 구성되어 있다. 이 중에서 <RDATA>는 리소스 레코드(RR) 데이터를 표시하는 부분으로써 각 유형별 타입마다 그 표시하는 정보에 적합하도록 다르게 정의된다. 단, 여기에서의 필드 순서는 이해의 편의를 위해 DNS 서버의 존 파일(zone file) 작성 시에 표기하는 순서를 따랐다. DNS 프로토콜에서의 리소스 레코드는 <RDLength>라는 <RRData>의 길이를 표시하는 필드를 추가하여 <Name><Type><Class><TTL><RDLength><RRData>의 순서구조를 갖는다.

<Name>은 도메인 네임을 의미한다. 도메인 네임은 필요한 리소스 레코드 데이터를 검색하기 위한 키 인덱스(key index) 역할을 한다. 리소스 레코드는 <Name>에 지정된 네임을 갖는 노드에 종속된 속성정보임을 의미한다.
<TTL>은 Time To Live 를 의미하는 필드이다. 이는 해당 리소스 레코드가 인터넷 상에 존재하는 리졸버의 캐시 테이블에 얼마나 오래 존속할 것인지를 결정하는 초(second) 단위의 값이다. 이 필드는 DNS 체계의 효율성을 위해 리졸버가 한번의 질의결과로 얻은 리소스 레코드 값을 캐시에 저장하여 일정기간 동안 관리함으로써 DNS 질의로 인한 과다 트래픽의 유발을 방지하는 데에 사용한다. 주로 인터넷 상에 존재하는 리졸버에 의해 참조된다.
<TTL>은 리소스 레코드별로 그 값이 정의된다. 따라서 일반적으로는 리소스 레코드 데이터가 자주 변경될 수 있는 리소스 레코드(RR)의 경우는 그 값을 적게, 그리고 네임서버 지정과 같이 오랫동안 바뀌지 않는 데이터를 지정하는 리소스 레코드(RR)의 경우는 길게 설정할 수 있다.
<TTL> 값은 0~2147483647의 값을 가질 수 있다. <TTL> 값이 0인 경우에는 리졸버는 해당 리소스 레코드를 캐시에 저장하지 않는다.

<Class>는 이제 그 구분의 의미가 없어진 필드로써 IN(Internet) 만을 사용하고 있다. BIND와 같은 경우에는 BIND DNS의 버전정보 등을 조회할 수 있도록 CHAOS 클래스를 사용하기도 한다.

<Type>은 각 리소스 레코드의 유형을 지정하는 필드이다. 리소스 레코드의 각 유형별 타입은 코드화 되어 있으며 이 코드는 각 레코드 타입별로 정의되어 있다. A 리소스 레코드(RR)는 1, NS 리소스 레코드(RR)는 2, SOA 리소스 레코드(RR)는 6 등으로 정의되어 있다. 이외에 다양한 리소스 레코드 <Type> 코드가 정의되어 있다.

<RDATA>는 각 리소스 레코드의 <Class>와 <Type> 별로 각기 다른 구조로 정의되었다. 호스트 주소에 대한 리소스 레코드(RR)에서 A type 리소스 레코드(RR)의 경우, <RDATA>는 32 bit로 고정된 데이터 공간을 가진다. 여기에 IPv4 주소가 이진(binary) 값으로 설정된다. IPv6가 도입되면서는 128 bit 크기로 고정된 데이터 공간을 지닌 AAAA 타입 리소스 레코드(RR)를 추가로 정의하였다. DNS 서버에 설정하는 존 파일(zone file)에서는 IP 주소를 텍스트 형식으로 작성하지만 DNS 질의에 의해 전달되는 IP 주소는 이진(binary) 형태의 데이터를 가진다.

NS 리소스 레코드의 경우에 <RDATA>는 권한을 지닌 네임서버의 도메인 네임을 지정한다.

□ 리소스 레코드 종류

도메인 네임에 대한 속성정보를 부여하기 위해 다양한 종류의 리소스 레코드가 정의되어 있다. 현재까지 정의된 리소스 레코드 종류는 아래와 같다.

TYPE	CODE	의미	비고
A	1	A host address	32bit IPv4 주소
NS	2	An authoritative name server	네임서버 도메인 네임 지정
MD	3	A mail destination (Obsolete-useMX)	사용하지 않음
MF	4	A mail forwarder(Obsolete-useMX)	사용하지 않음
CNAME	5	The canonical name for an alias	Alias 도메인 네임 설정, 원래 도메인 네임을 매핑
SOA	6	Marks the start of a zone of authority	Zone의 속성 정보 지정
MB	7	A mailbox domain name(EXPERIMENTAL)	시험적 구현
MG	8	A mail group member(EXPERIMENTAL)	시험적 구현
MR	9	A mail rename domain name(EXPERIMENTAL)	시험적 구현
NULL	10	A null RR(EXPERIMENTAL)	시험적 구현
WKS	11	A well known service description	호스트의 특정 IP주소 및 해당 IP 주소를 통해 제공하는 TCP/UDP 서비스 포트정보 지정
PTR	12	A domain name pointer	도메인 네임을 매핑함 주로 IP 주소의 도메인네임 지정에 사용
HINFO	13	Host information	호스트의 CPU와 OS 정보
MINFO	14	Mailbox or mail list information (EXPERIMENTAL)	시험적 구현
MX	15	Mail exchange	메일서버의 도메인 네임 지정
TXT	16	Text strings	문자열 정보를 지정
RP	17	For Responsible Person	도메인 네임별 담당자 정보
AFSDB	18	For AFS DataBase location	AFS DB 위치정보
X25	19	For X.25 PSDN address	도메인 네임의 X.25 주소정보
ISDN	20	For ISDN address	도메인 네임의 ISDN 주소정보
RT	21	For Route Through	
NSAP	22	For NSAP address	NSAP 주소정보
NSAP-PTR	23		
SIG	24	For security signature	보안 서명을 저장
KEY	25	For security key	보안 키를 저장
PX	26	X.400 mail mapping information	
GPOS	27	Geographical Position	도메인 네임의 지리적 위치정보 위도, 경도, 고도

AAAA	28	IP6 Address	128 bit IPv6 주소정보
LOC	29	Location Information	
NXT	30	Next Domain	
EID	31	Endpoint Identifier	
NIMLOC	32	Nimrod Locator	
SRV	33	Server Selection	
ATMA	34	ATM Address	
NAPTR	35	Naming Authority Pointer	
KX	36	Key Exchanger	
CERT	37	CERT	
A6	38	A6 (IPv6 Address)	IPv6 Prefix 및 주소정보 사용하지 않을 예정
DNAME	39	DNAME	A6과 함께 사용하는 도메인 매핑 정보 사용하지 않을 예정
SINK	40	SINK	
OPT	41	OPT	
APL	42	APL	
SSHFP	44	SSH Key Finger print	
UINFO	100		
UID	101		
GID	102		
UNSPEC	103		
TKEY	249	Transaction Key	DNS 질의에만 사용
TSIG	250	Transaction Signature	DNS 질의에만 사용
IXFR	251	Incremental transfer	DNS 질의에만 사용
AXFR	252	Transfer of an entire zone	DNS 질의에만 사용
MAILB	253	mailbox-related RRs (MB, MG or MR)	DNS 질의에만 사용
MAILA	254	Mailagent RRs (Obsolete - see MX)	DNS 질의에만 사용
*	255	A request for all records	DNS 질의에만 사용

[표 1-2] 리소스 레코드(Resource Record) 종류

상당히 많은 종류의 리소스 레코드가 정의되었지만 이 중에서 주로 사용하는 리소스 레코드는 한정되어 있다. 그러나 아래의 다양한 리소스 레코드를 용도에 적합하게 사용함으로써 DNS 도메인 데이터베이스 체계를 효율적으로 사용할 수 있다.

위 리소스 레코드 중에서 그 코드가 상위에 정의된 리소스 레코드들은 실제 도메인 데이터베이스를 구성하는 리소스 레코드가 아닌 DNS 질의에만 사용하는 가상적인 리소스 레코드 타입(type)이다. 예를 들면 AXFR 타입을 사용한 DNS 질의에 대해서는 해당 도메인 네임을 도메인 존(zone)으로 하는 존(zone)의 전체 리소스 레코드(RR)를 모두 응답한다. * 의 경우에는 해당 도메인 네임이 가지는 모든 종류의 리소스 레코드(RR) 들을 응답한다. TSIG 및 TKEY는 보안이 적용된 DNS 질의응답을 수행하기 위해 사용한다.

□ 도메인 위임 관련 리소스 레코드

리소스 레코드 중에서 도메인 네임 체계의 위임체계와 관련된 정보를 표현하는 특별한 리소스 레코드(RR)가 있다. 이에는 SOA(Start of a zone Of Authority) 리소스 레코드(RR)와 NS(authoritative Name Server) 리소스 레코드(RR)가 있다.

SOA 리소스 레코드(RR)는 도메인 존(zone)에 대한 정보를 표시한다.
SOA 리소스 레코드는 다른 리소스 레코드와 동일하게 다음과 같은 구조를 갖는다.
<Name><TTL><Class><Type=SOA><RDATA>

단, 여기에서의 <Name>은 존(zone)의 도메인 네임이다. 즉, 상위 계층의 도메인에서 위임을 받은 노드의 도메인 네임이어야 한다. SOA 리소스 레코드는 존(zone)에 대한 정보를 지정하는 리소스 레코드이므로 존(zone)에 속한 다른 일반 도메인 네임은 SOA 레코드를 소유할 수 없다.

SOA 리소스 레코드의 <RDATA>는 다음과 같은 내부 필드 구조를 갖는다.
<MNAME><RNAME><SERIAL><REFRESH><RETRY><EXPIRE><MINIMUM>

<MNAME>은 해당 존(zone)의 최상위 마스터 네임서버(primary master name server) 도메인 네임을 지정한다. 이때 <MNAME>의 도메인 네임은 FQDN으로 표기한다. <MNAME>에 지정된 네임서버는 해당 존(zone)에 대한 전적인 관리 권한을 지닌 마스터 네임서버임을 의미한다. 이후에 도메인의 위임과 관련된 장에서 관련된 네임서버의 마스터 네임서버, 슬레이브 네임서버, 최상위 네임서버에 관련한 상세한 설명을 한다.
<MNAME>에 지정된 최상위 마스터 네임서버는 해당 도메인의 존(zone) 데이터의 생성, 변경, 삭제 등을 수행할 수 있는 관리 권한을 지니는 네임서버이다. 이 네임서버는 후에 DNS UPDATE와 같은 도메인 네임 및 리소스 레코드의 동적 갱신을 수행하는 메커니즘에서 참조되며, 또한 마스터 네임서버와 슬레이브 네임서버간에 도메인 존(zone) 데이터 전체를 전송하는 메커니즘에 있어 중요한 역할을 한다.

<RNAME>은 해당 존(zone)을 관리하는 담당자(Responsible Person)의 이메일(email) 주소를 표기하는 필드이다. 지정된 도메인 존(zone)의 담당자의 정보를 제공하는 역할을 한다.

<SERIAL> 필드는 해당 존(zone)의 변경에 따른 버전번호 정보 필드이다. 존(zone) 내부의 데이터가 변경될 때마다 이 <SERIAL> 필드의 버전 번호는 하나 이상으로 증가되어야 한다. <SERIAL> 필드의 버전 번호는 해당 존(zone)의 변경여부를 파악할 수 있는 중요한 정보를 제공한다. 마스터 네임서버와 슬레이브 네임 서버간에 존(zone) 데이터의 일치를 유지하기 위한 메커니즘에서 존(zone)의 변경여부를 확인하기 위해 참조된다. 수작업으로 그 변경 작업을 하는 경우에는 일반적으로 YYYYMMDDnn 의 형식으로 버전 번호를 관리한다. 여기에서 YYYY=년도, MM=월, DD=일, nn=일련번호의 의미를 지닌다. 이 경우 사용할 수 있는 연도는 4294 까지이다. 그러나 도메인 네임과 리소스 레코드에 대한 동적변경 메커니즘으로 DNS UPDATE를 사용하는 경우, 이 <SERIAL> 필드는 마스터 네임서버에 의해 자동으로 관리된다.

<REFRESH>와 <RETRY>, <EXPIRE> 필드는 이 존(zone)에 대한 정보를 갱신하는 메커니즘에 있어서의 타이머 시간 정보를 지정한다. 슬레이브 네임서버는 <REFRESH>에 지정된 타이머 시간이 경과한 후에 해당 존(zone)에 대한 갱신을 시도한다. 즉, <REFRESH>에 지정된 시간은 슬레이브 네임서버가 해당 존(zone)에 대한 변경여부를 확인하는 주기로 설정된다. 슬레이브 네임서버는 <REFRESH>에 지정된 주기로 마스터 네임서버에 대하여 대상 존(zone)의 SOA 리소스 레코드를 질의하여 <SERIAL> 버전이 자신이 알고 있는 <SERIAL> 버전을 기준으로 갱신여부를 확인한다.

<RETRY> 필드 값은 <REFRESH>에 의한 갱신 시도가 실패한 경우, 다시 존(zone)에 대한 갱신을 시도하는 타이머 시간의 의미를 지닌다. <RETRY> 필드 값은 일반적으로 <REFRESH> 값보다 작은 값이어야 한다. <REFRESH> 주기에 의한 존(zone) 버전정보의 확인을 하기 이전에 보다 짧은 시간 내에 재시도를 하도록 하기 위함이다.

<EXPIRE> 필드는 슬레이브 네임서버가 최상위 마스터 네임서버로 대상 존(zone)에 대한 갱신여부 확인에 실패한 경우 슬레이브에 존재하는 동일 존(zone)의 복사 존(zone)을 유효한 것으로 유지할 것인가에 대한 시간 설정값이다. <EXPIRE> 필드에 지정된 기간 동안 슬레이브 네임서버가 최상위 마스터 네임서버로부터 대상 존(zone)에 대한 갱신여부를 확인할 수 없는 경우, 슬레이브 네임서버는 이 존(zone) 전체를 유효하지 않은 것으로 판단하고 이 존(zone)에

속한 모든 도메인 네임에 대한 DNS 응답을 중지한다. 그러나 존(zone)에 대한 네임 서비스(name service)를 중지한 이후에도 <REFRESH>에 설정된 주기로 해당 존(zone)의 갱신여부를 최상위 마스터 네임서버에 대해 확인하는 시도를 지속한다. <EXPIRE> 값은 일반적으로 <RETRY>나 <MINIMUM> 값보다 큰 값으로 설정하며 일반적으로 권장하는 설정값은 2주~4주 정도이다.

<MINIMUM> 필드는 이 존(zone)에 속한 모든 리소스 레코드의 디폴트 TTL (default TTL) 값을 지정한다. 이 값은 최소 TTL 값을 지정하는 것으로 만일 리소스 레코드에 TTL 값이 관리자에 의해 지정되지 않은 경우, 이 <MINIMUM> 필드의 최소 TTL 값을 지정하여 사용한다. TTL 값은 리졸버(resolver)의 캐시(cache)에 리소스 레코드를 저장, 존속시키는 기간으로 사용된다. <MINIMUM> 필드의 값으로는 약 1~5일의 기간을 설정하는 것이 권장되고 있다. 단, 이 <MINIMUM> 필드의 값은 리소스 레코드의 TTL이 명시적으로 지정된 경우, 리소스 레코드별 명시적인 TTL 지정값으로 대체하여 사용한다.

이와 같이 SOA 레코드는 존(zone)의 관리에 필요한 정보를 갖고 있는 중요한 레코드이다. 그리고 반드시 하나의 존(zone)에 반드시 하나의 SOA 레코드만 지정되어 있어야 한다.

NS 리소스 레코드(RR)는 특정 도메인 존(zone)이 어느 네임서버에 위임되어 설정되어 있는지를 표시한다.

NS 리소스 레코드도 다른 리소스 레코드와 동일하게 다음과 같은 구조를 갖는다.
<Name><TTL><Class><Type=NS><RDATA>

이 중에서 <Name>은 위임되는 또는 위임된 존(zone)의 도메인 네임을 지정한다.

NS 리소스 레코드의 <RDATA>는 NS 리소스 레코드의 목적에 따라 아래와 같이 NS 레코드의 고유한 구조를 갖는다.
<NSDNAME>

NS 레코드의 경우, <RDATA> 영역에 단일한 필드 <NSDNAME>만을 가지고 있다. <NSDNAME>은 NS 레코드를 소유한 <Name>의 도메인 존(zone)을 가지고 있는 네임서버(name server)의 도메인 네임을 지정하는 필드이다. 이 필드는 FQDN의 도메인 네임을 설정한다.

NS 레코드는 SOA 레코드와 마찬가지로 도메인 존(zone)에 대한 정보를 지니고 있다. 단, NS 레코드는 해당 존(zone)이 실제로 위치한 네임서버를 도메인 네임으로 지정하는 정보를 지니고 있다.

NS 레코드는 그 사용되는 방식에 따라 다음의 2가지 의미로 사용한다.

첫째는 상위 도메인 존(zone)에서 위임되는 도메인 존(zone)의 네임서버를 지정하는 경우이다.
예를 들어 example.co.kr. 존(zone)은 상위 도메인 co.kr. 존(zone)에서 위임설정이 되어 있다. 상위 도메인 co.kr.의 존(zone) 내부에는 위임 설정을 위해 아래와 같은 도메인 네임에 대한 리소스 레코드를 정의한다.

<Name=example.co.kr.><TTL><Class><Type=NS>< NSDNAME=ns1.example.co.kr.>
<Name=example.co.kr.><TTL><Class><Type=NS>< NSDNAME=ns2.example.co.kr.>
<Name=ns1.example.co.kr.><TTL><Class><Type=A><ADDRESS=192.0.2.11>
<Name=ns1.example.co.kr.><TTL><Class><Type=A><ADDRESS=192.0.2.12>

이는 co.kr. 도메인에서 example.co.kr. 이라는 서브 도메인(sub domain)이 ns1.example.co.kr.과 ns2.example.co.kr.의 네임서버에 위치하고 있음을 설정한 것이다. 이러한 설정은 리졸버 서버가 루트 도메인으로부터 example.co.kr. 도메인에 대한 질의를 하는 과정에서 co.kr. 도메인에 이르러 원하는 도메인 example.co.kr.이 설정된 네임서버 정보를 얻을 수 있게 한다.
곧, 도메인을 위임하는 상위 도메인에 설정하는 NS 레코드는 리졸빙(resolving) 과정에서 도메인 트리 구조에서 해당 도메인을 검색하기 위한 다음 단계 네임서버의 위치 정보를 알려주는 역할을 한다.

이 경우에 NS 레코드에 의해 지정된 네임서버의 도메인 네임에 대해 네트워크 주소 정보의 지정이 필요하다.

리졸버가 co.kr. 도메인에 대해 질의한 결과 그 네임서버의 도메인을 파악했다 하더라도 네임서버의 도메인이 example.co.kr. 도메인에 속해있다면 아직 그 네임서버의 네트워크 주소를 알지 못하는 상황에서는 도메인에 대한 질의절차는 물론 네임서버의 네트워크 주소를 파악할 수가 없게 된다. 이러한 상황을 피하기 위해 상위 도메인이 위임설정을 하는 경우, 해당 네임서버의 도메인 네임을 NS 레코드로 지정한 후 해당 도메인 네임이 자신의 도메인에 속한 도메인 네임일

경우, 즉 위 예시와 같이 example.co.kr. 도메인 네임이 co.kr. 도메인에 속해 있는 경우에는 해당 도메인 네임의 네트워크 주소정보를 제공해야 한다.
NS 레코드 아래에 지정된 A 리소스 레코드는 이를 위한 것이며 이를 글루 레코드라 한다.

글루 레코드(glue record)는 그 단어가 의미하는 바와 같이 접착제 역할을 한다. 곧, 상위 도메인에서 하위 도메인으로의 위임관계를 위해 하위 도메인의 네임서버로 네트워크 주소를 알려줌으로써 네임 리졸빙(name resolving) 절차를 연결해 주는 역할을 한다.
상위 도메인의 글루 레코드 설정을 위해 상위 도메인과 동일 도메인에 속한 도메인 네임을 갖는 네임서버를 사용하는 경우, 도메인 신청시 해당 네임서버의 도메인 네임에 대한 네트워크 주소정보를 함께 알려주어야 한다.
글루 레코드는 리졸버에 의한 리졸빙 절차에 사용되므로 만일 글루 레코드의 네트워크 주소가 실제 사용하는 네임서버의 네트워크 주소와 상이하다면 리졸빙에 문제가 발생할 수 있다.

두 번째는 해당 존(zone)의 SOA 레코드와 함께 해당 존(zone)에 대해 지정된 NS 레코드의 경우이다.
예로써 example.co.kr. 도메인은 위의 co.kr. 도메인으로부터 위임을 받아 ns1.example.co.kr. 네임서버와 ns2.example.co.kr. 네임서버에 아래와 같이 리소스 레코드를 설정한다.

<Name=example.co.kr.><TTL><Class><Type=SOA><MNAME=ns1.example.co.kr.><RNAME><SERIAL><REFRESH><RETRY><EXPIRE><MINIMUM>
<Name=example.co.kr.><TTL><Class><Type=NS><NSDNAME=ns1.example.co.kr.>
<Name=example.co.kr.><TTL><Class><Type=NS><NSDNAME=ns2.example.co.kr.>
<Name=ns1.example.co.kr.><TTL><Class><Type=A><ADDRESS=192.0.2.11>
<Name=ns2.example.co.kr.><TTL><Class><Type=A><ADDRESS=192.0.2.12>

이때 설정된 NS 레코드는 이 도메인 존(zone)의 네임서버를 지정하는 것으로 리졸버의 질의절차 보다는 도메인 존(zone)에 대한 관리 메커니즘에 사용된다.
위의 예에서 SOA 레코드에 지정된 <MNAME=ns1.example.co.kr.>은 이 도메인 존(zone)의 최상위 마스터 네임서버가 ns1.example.co.kr. 네임서버 임을 지정한다. 그리고 그 아래 2개의 NS 레코드는 각각 ns1.example.co.kr.과 ns2.example.co.kr. 서버가 이 도메인 존(zone)의 네임서버 임을 지정한다.

SOA 레코드의 <MNAME>과 해당 존(zone)의 NS 레코드를 통해 이 도메인 존(zone)에 대한 네임서버 구성정보를 파악한다. 즉, 위와 같은 경우에는 이 도메인 존(zone)은 2개 네임서버 ns1.example.co.kr.과 ns2.example.co.kr.이 존재하며 이 중에서 ns1.example.co.kr.이 마스터 네임서버(master name server)이고 ns2.example.co.kr.이 슬레이브 네임서버로 설정되어 있는 구조임을 알려준다.
이러한 정보는 해당 도메인 존(zone)이 생성되거나 수정이 가해지는 경우에 네임서버 중에서 어느 네임서버가 슬레이브 네임서버로써 존(zone) 데이터의 전송을 마스터 네임서버로 요청해야 하는 지를 알려준다.
곧, 도메인 존(zone)에 설정된 존(zone)의 도메인 네임에 대한 NS 레코드는 SOA 레코드와 더불어 도메인 존(zone)에 대한 관리 메커니즘을 정의하는 역할을 한다.

이 정보들은 마스터 서버가 특정 도메인 존(zone)의 갱신이 이루어진 경우, 슬레이브 서버로 이를 알리는 NOTIFY 메시지를 전송해야 할 때, 대상 존(zone)의 NOTIFY 메시지를 수신해야 하는 네임서버 정보를 파악하는데 참조된다. 이 경우, NS 레코드로 지정된 네임서버 중 최상위 마스터 네임서버에 해당하는 네임서버를 제외한 모든 NS 레코드의 지정된 네임서버로 NOTIFY를 송출한다.

AXFR 및 IXFR 등 존(zone) 데이터의 전송(transfer) 메커니즘에서는 슬레이브 서버가 AXFR 또는 IXFR 메시지와 같은 전송요청 메시지를 송출해야 할 최상위 마스터 네임서버를 파악하는 데에 있어 SOA 레코드의 <MNAME> 필드 값을 참조한다.
이외에 도메인 네임과 리소스 레코드에 대한 DNS 동적변경(dynamic update) 메커니즘, 즉 DNS UPDATE에서는 수정을 하고자 하는 도메인 존(zone)의 SOA 레코드를 파악한 후 네임서버 중에서 해당 리소스 레코드 수정이 이루어져야 하는 최상위 마스터 네임서버로 직접 접근하여 해당 레코드의 동적변경을 수행한다.

이상과 같이 SOA 레코드와 NS 레코드는 주로 도메인의 위임구조 및 도메인 관리를 위한 메커니즘을 구현함에 활용한다.
따라서 도메인 존(zone)의 이 SOA 레코드와 NS 레코드의 설정에는 각별한 주의를 기울이는 것이 필요하다.

□ 호스트 주소 리소스 레코드

호스트 주소(Host Address) 리소스 레코드는 가장 일반적으로 광범위하게 사용되는 리소스 레코드로써 리소스 레코드 타입은 A(Address)로 표기된다. A 리소스 레코드는 IPv4 주소 정보를 위한 레코드이다. IPv6의 도입으로 인해 새로운

리소스 레코드 AAAA 리소스 레코드가 새롭게 추가 정의되었다. 이로써 현재 인터넷을 위한 호스트 주소 리소스 레코드는 A 리소스 레코드와 AAAA 리소스 레코드가 존재한다.
A 리소스 레코드는 아래와 같은 구조를 지닌다.

<Name><TTL><Class=IN><Type=A><RDATA>

<Name> 필드는 임의의 도메인 네임이 될 수 있다. A 리소스 레코드를 가지는 도메인 네임을 특히 호스트 네임(Host Name)이라 한다.

<Class>는 인터넷(IN, Internet)만이 가능하다. 일반적인 리소스 레코드와는 달리 A 리소스 레코드는 인터넷 클래스에 대해서만 정의된다.

<RDATA>는 A 리소스 레코드의 특성에 맞게 아래와 같은 구조를 지닌다.
<ADDRESS>

A 리소스 레코드는 단일 필드 <ADDRESS> 만을 가진다.
<ADDRESS> 필드는 32 bit의 크기를 갖는 필드로써 IPv4 헤더의 주소 필드와 같은 형식을 사용한다. 즉, 네트워크 상에서 사용하는 이진(binary) 32 bit IP 주소를 여기에 설정한다. 네임서버의 존 파일(zone file)에는 텍스트 형태의 IP 주소 표기를 사용하여 설정한다.

AAAA 리소스 레코드는 아래와 같은 구조를 가진다.

<Name><TTL><Class=IN><Type=AAAA><RDATA>

<Name>은 A 레코드와 동일하게 임의의 도메인 네임이며 호스트 네임이다. 다만 IPv6 주소를 갖는 호스트 도메인 네임이 설정될 수 있다.

<Class>는 A 레코드의 경우와 동일하게 인터넷(IN, Internet)만이 가능하다.

AAAA 리소스 레코드는 <RDATA> 영역에 단일 주소 필드로 정의되어 있다.
주소(address) 필드는 128bit 크기로 IPv6 헤더의 주소 필드와 동일한 형식의 네트워크 바이트 순서(network-byte order)로 설정된 데이터를 설정한다.

특정 도메인 네임에 대한 A 타입의 질의는 해당 도메인이 소유한 A 리소스 레코드 정보만 응답된다. 또한 AAAA 타입의 질의에 대해서는 해당 도메인 네임이 소유한 AAAA 리소스 레코드 정보만 응답된다.
따라서 임의의 도메인 네임이 IPv4 또는 IPv6 주소를 갖고 있는지의 여부는 A 타입의 질의와 AAAA 타입의 질의를 모두 수행함으로써만 파악할 수 있다.

□ 이메일 서비스 관련 리소스 레코드

인터넷 네트워크 주소 정보이외에 가장 많이 사용하고 있는 리소스 레코드로는 MX 리소스 레코드가 있다. MX는 메일 교환기(Mail Exchange)를 의미하며 인터넷 상에서는 메일 서버를 의미한다.

메일 서비스를 위해서는 메일 주소가 필요한데 이메일(email)의 주소체계는 <user account>@<domain name>의 형식을 갖는다. 이러한 이메일 주소체계를 DNS를 사용하여 효율적으로 메일 라우팅이 가능하도록 하기 위해 사용하는 리소스 레코드가 MX 레코드이다.

MX 리소스 레코드는 아래와 같은 구조를 갖는다.

<Name><TTL><Class><Type=MX><RDATA>

<Name>은 도메인 네임을 지정한다. 흔히는 회사의 도메인 네임, 즉 도메인 존(zone) 네임에 지정한다. 그러나 존(zone)에 속하는 임의의 도메인 네임을 <Name>으로 지정할 수 있다.
도메인 example.co.kr.의 경우, 도메인 존(zone) 네임인 example.co.kr. 도메인 네임에 대해 MX 레코드를 설정할 수 있다. 또한 sales.example.co.kr. 이라는 영업부서용 도메인 네임을 설정하여 이에 대해 MX 레코드를 설정할 수도 있다.

MX 레코드의 <RDATA>는 다음과 같은 구조를 갖는다.
<PREFERENCE><EXCHANGE>

<PREFERENCE>는 우선 순위를 지정하는 필드로써 뒤에 지정되는 메일 서버에 대한 선호도를 지정한다. 이 필드의 값에 따라 외부에서 이 사이트로 메일을 전송하는 메일 서버는 가장 값이 적은 <PREFERENCE> 필드 값을 갖는 리소스

레코드의 메일서버 도메인 네임을 선택한다. 이에 따라 다수의 메일 서버가 존재할 수 있으며 이들 사이에 <PREFERENCE> 필드를 사용하여 우선순위를 설정하는 것이 가능하다.

<EXCHANGE>는 메일서버의 도메인 네임을 지정한다.

MX 레코드에 대한 질의는 MX 레코드와 MX 레코드에 지정된 메일서버 도메인 네임의 IP 주소를 추가 정보로 응답한다.

아래는 example.co.kr. 도메인의 MX 레코드 설정 예이다.

<Name=example.co.kr.><TTL><Class><Type=MX><PREFERENCE=10><EXCHANGE=mail.example.co.kr.>

<Name=mail.example.co.kr.><TTL><Class=IN><Type=A><ADDRESS=192.0.2.21>

1.5 DNS 네임서버 및 리졸버 개요

DNS는 크게 다음의 3가지 요소로 구성된다.

도메인 네임 공간(Domain Name Space)과 리소스 레코드(Resource Record)
네임서버(Name Server)
리졸버(Resolver)

이 중에서 도메인 네임 공간(Domain Name Space)과 리소스 레코드(Resource Record)는 앞장에서 설명한 바와 같다.

도메인 네임 공간은 도메인 네임의 구성과 그 구조를 정의하는 반면, 리소스 레코드는 하나의 도메인 네임이 가질 수 있는 속성 정보 군을 정의한다.
도메인 네임은 도메인 네임 공간으로부터 도메인 할당기관의 관리를 통해 할당과 위임을 거쳐 이를 필요로 하는 기관 또는 개인에게 부여된다.
리소스 레코드는 도메인 존(zone)을 기준으로 관리권한을 가진 영역내의 도메인 네임에 대한 속성정보를 필요에 따라 생성, 유지, 관리할 수 있는 도메인 데이터베이스를 형성한다.

이 도메인 데이터베이스는 인터넷 통신에 활용될 수 있는 형태로 구체화될 필요성이 있는데, 이러한 기능을 담당하는 구성 요소가 네임서버와 리졸버다.

네임서버는 도메인 데이터베이스의 일정 영역(zone)을 소유하고 있는 DNS 서버이다. 도메인 데이터베이스는 DNS 체계에 따라 분산구조를 지니고 있다. 네임서버는 이 전체 도메인 데이터베이스 중에서 관리 권한을 위임받은 일정 영역(zone)의 데이터베이스를 관리하고 유지하며 또한 인터넷 상의 임의의 리졸버로부터의 데이터 요청에 대해 응답한다. 즉, 네임서버는 도메인 데이터베이스 서버 역할을 담당한다고 할 수 있다.

네임서버는 다음의 2가지의 모드로 동작할 수 있다.

리커시브가 아닌 모드(non-recursive mode)
리커시브 모드(recursive mode)

리커시브가 아닌 모드(non-recursive mode)는 네임서버가 기본적으로 동작하는 동작 모드로써 모든 네임서버는 이 동작 모드를 구현해야 한다. 리커시브가 아닌 모드에서 네임서버는 자신이 갖고 있는 도메인 데이터베이스 영역(zone)의

정보에 대해서만 권한을 지니고 응답한다. 그 외의 다른 도메인 중 자신이 소유하고 있는 도메인으로부터 위임된 도메인에 대해서는 가장 근접한 네임서버의 정보를 참조정보로 응답한다.

리커시브 모드(recursive mode)는 네임서버가 옵션기능으로 구현, 동작할 수 있는 동작 모드이다. 이는 주로 클라이언트 호스트의 리커시브 질의(recursive request) 요청에 부응하기 위한 것이다. 리커시브 모드로 동작하는 네임서버는 클라이언트에서 리커시브 질의 요청이 있는 경우, 자신의 포함하고 있는 리졸버 루틴을 사용하여 리졸버의 질의절차를 수행하고 최종 응답결과를 클라이언트로 응답한다. 곧, 리커시브 네임서버는 본래의 네임서버 기능에 리졸버의 역할 기능까지 포함하여 동작하는 네임서버라 할 수 있다. 이러한 리커시브 모드로 동작하는 네임서버를 리커시브 네임서버(recursive name server)라 한다.

리졸버는 도메인 데이터베이스로부터 클라이언트 프로그램이 요청하는 정보를 조회하여 클라이언트 프로그램에게 제공하는 기능을 수행한다. 곧, 리졸버는 도메인 네임서버와 사용자 프로그램(user program)간의 인터페이스의 기능을 한다. 리졸버는 수많은 네임서버에 의해 구성된 전체 도메인 데이터베이스에서 원하는 정보가 있는 위치를 파악하고 해당 네임서버에 접근하여 요청된 정보를 조회하는 기능을 수행할 수 있어야 한다.

그러나 이러한 리졸버의 전 기능을 클라이언트 호스트에 구현하는 것은 단말 시스템의 시스템 자원의 한계 등으로 비현실적인 면이 있다. 이에 따라 리졸버의 대부분의 기능을 리커시브 네임서버로 이전하고 단말 호스트 시스템에는 단순한 기능만을 지닌 리졸버 루틴을 구현하는 옵션이 제시되었다. 이러한 단순화된 기능의 리졸버를 스터브 리졸버라 한다.

스터브 리졸버(stub resolver)는 현재 대부분의 호스트에 구현된 OS 시스템의 리졸버 루틴으로 구현되어 있다. 스터브 리졸버는 리졸버의 기능 대부분을 리커시브 네임서버에 의존하므로 매우 간단한 동작만을 수행한다. 스터브 리졸버는 전체 도메인 데이터베이스의 구조를 파악할 필요 없이 리커시브 네임서버 역할을 해 줄 네임서버의 IP 주소만 파악하면 된다. 이후의 도메인 데이터베이스에 대한 질의는 리커시브 질의요청을 포함한 DNS 질의를 지정된 리커시브 네임서버로 전송하고 최종 결과를 네임서버로부터 응답 받는 절차만 수행한다. 대부분의 단말에 설정하는 DNS 서버는 이 리커시브 네임서버를 의미하는 것으로 주로 ISP 등에서 사용자들을 위해 운영하고 있는 리커시브 모드의 동작을 하도록 설정된 네임서버이다. 스터브 리졸버는 간단한 기능의 리졸버로 각 시스템의 시스템 루틴으로

존재하면서 사용자 프로그램의 도메인 네임의 IP 주소로의 변환 요청을 처리하는 인터페이스 기능을 수행한다.

스터브 리졸버에 반하여 리커시브 네임서버는 전체 도메인 데이터베이스에 대한 접근이 가능하여야 한다. 리졸버는 루트 노드를 시작으로 도메인 루트 구조를 순차적으로 검색해야 하므로 루트 네임서버의 IP 주소를 그 초기 값으로 가져야 할 필요성이 있다. 따라서 리커시브 네임서버를 구성하는 경우, 네임서버의 환경 설정 파일에 루트 네임서버의 IP 주소 정보파일을 수작업으로 설정해 주어야 한다. 루트 네임서버의 IP 주소 정보 파일은 INTERNIC에서 관리를 하고 또한 인터넷 상으로 제공하고 있는 ftp://ftp.rs.internic.net/domain/named.cache 의 파일을 다운받아 설정한다.

1.6 DNS 도메인 위임설정 개요

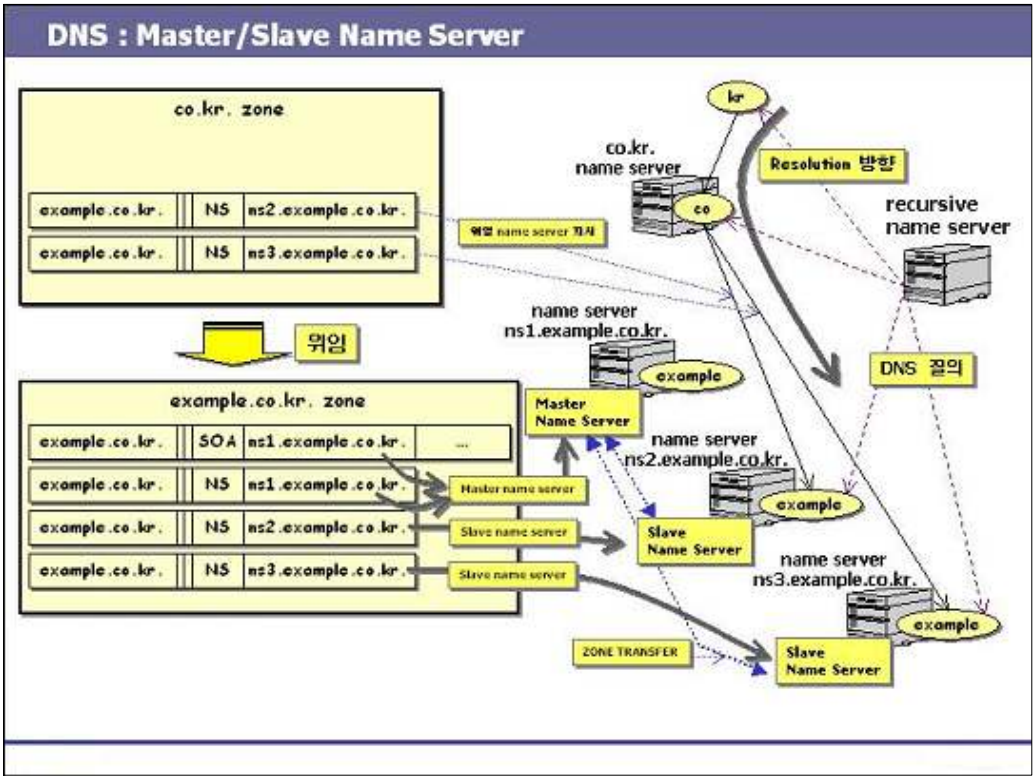
DNS는 분산된 도메인 데이터베이스를 기본구조로 한다고 이미 말한 바 있다.

리졸버를 통한 도메인 데이터베이스에 대한 조회가 가능하기 위해서는 도메인 데이터베이스가 인터넷 상에 안정적으로 구성이 되어 있어야 한다.

트리 구조의 도메인 네임체계와 이와 관련된 리소스 레코드들이 인터넷 상에서 분산 데이터베이스를 형성하는데 있어 위임(delegation) 체계는 중요한 요소가 된다.

여기에서는 주로 도메인의 위임과 관련된 구성이 어떻게 이루어지는가에 그 초점을 둔다.

도메인의 위임은 도메인 존(zone)에 해당하는 노드에 대해 설정한다.



[그림 1-6] DNS 마스터/슬레이브 네임서버 위임구성 사례 -1

그림은 co.kr. 도메인에서 도메인 존(zone) example.co.kr. 이 위임된 상황을 보인 것이다.

도메인 co.kr.에서는 example 노드를 생성하고 이 노드의 도메인 네임에 대해 NS 리소스 레코드를 설정한다.
이때, 만일 example.co.kr. 도메인이 3개의 네임서버를 사용한다고 가정할 때, 각각을 NS 레코드를 사용하여 설정한다. 상위 도메인인 co.kr. 도메인에서의 위임되는 도메인 네임에 대해 NS 레코드를 지정하는 것은 해당 네임서버로 접근하는 경우 위임되는 도메인의 SOA 레코드 정보를 제공받을 수 있다는 것을 함축적으로 의미한다. 곧 NS 레코드로 설정되는 도메인 네임은 하위 도메인 존(zone)의 도메인 네임이라는 의미를 가지고 있다.

상위 도메인 co.kr.의 NS 레코드는 리졸버에 의해 사용된다. 리졸버는 도메인 위임 트리 구조를 루트 도메인으로부터 하위 도메인으로 탐색하면서 도메인 co.kr의 네임서버에서 example.co.kr. 도메인의 네임서버 정보를 얻는다. 위 경우에는 3개의 네임서버에 대한 정보를 얻게 된다.
리졸버의 입장에서는 example.co.kr 도메인의 네임서버들을 정확히 파악하여 원하는 도메인 네임의 정보를 얻을 수 있는 것으로 충분하다. 상위 도메인의 특정 도메인에 대한 NS 레코드를 사용한 위임설정은 도메인 트리 구조를 형성하는 수단으로써 리졸버의 도메인 네임 리졸루션(resolution) 절차상에서 원하는 도메인 네임의 정보가 소재한 네임서버의 위치를 제공하는 결정적인 역할을 한다.

상위 도메인에서 위임된 하위 도메인의 네임서버에 대한 NS 레코드 정보만으로는 리졸버가 참조된 다음 네임서버로 네트워크 접근을 하는 데에 정보가 불충분할 수 있다. 이는 NS 리소스 레코드는 네임서버의 도메인 네임만을 제공하기 때문이다. 이러한 이유로 상위 도메인에서는 위임하는 하위 도메인의 네임서버에 대한 A 레코드 정보를 자신의 도메인 존(zone)에서 관리한다. 이로써 리졸버의 DNS 질의에 대해 해당 도메인의 네임서버와 그 네임서버의 IP 주소를 포함한 응답을 함으로써 리졸버가 다음 네임서버로 접근하는 데 충분한 정보를 제공한다. 이때 하위 도메인의 네임서버에 대한 A 리소스 레코드를 글루 레코드라 한다.

이 글루 레코드(glue record)는 해당 네임서버의 도메인 네임이 위임을 하는 상위 도메인에 속한 도메인 네임인 경우에 한한다. 곧, 앞의 그림에서와 같이 example.co.kr. 도메인의 네임서버가 ns1.example.co.kr. 인 경우에는 co.kr. 도메인에 속한 도메인 네임이므로 co.kr. 도메인 존(zone)에 그 글루 레코드를 설정한다. 그러나 만일 네임서버가 ns1.example.com.과 같이 com. 도메인에

속한 도메인 네임일 경우에는 글루 레코드를 co.kr. 도메인 존(zone)에 설정할 수 없다. 이런 경우에는 example.com. 도메인에 ns1.example.com. 도메인 네임에 대한 A 리소스 레코드를 반드시 설정해 주어야 한다. 만일 이 리소스 레코드의 설정이 누락되어 있다면 리졸버가 example.co.kr.의 네임서버로 접근할 수 없는 상황이 발생하여 도메인 정보를 얻을 수 없게 된다.

한편 example.co.kr. 도메인의 관리자의 입장에서는 도메인 존(zone)에 대한 생성, 변경, 삭제 등의 관리를 안정적으로 할 수 있는 방안이 필요하다. example.co.kr. 도메인의 경우, 그 네임서버로서 3개의 네임서버를 구성하고 있다. 이는 네임서버를 안정적으로 구성하기 위한 것이다. 그러나 2개 이상의 네임서버를 구성하는 경우, 각 네임서버의 도메인 존(zone)에 대한 관리에 있어 모든 네임서버의 도메인 존(zone) 내용이 동일하도록 유지할 필요성이 발생한다. 즉, 도메인 존(zone) 내용이 어느 네임서버를 통해서 DNS 질의/응답을 수행하더라도 동일하도록 관리해야 할 필요성이 있다.

마스터 네임서버(master name server)와 슬레이브 네임서버(slave name server)의 구성은 이러한 관리상의 목적을 충족시키기 위한 구성방법이다. 마스터 네임서버를 주 네임서버(primary name server), 슬레이브 네임서버를 보조 네임서버(secondary name server)라고 하기도 한다.

마스터 네임서버와 슬레이브 네임서버의 구분은 대상 도메인 존(zone) 단위로 기준하여 구분한다. 하나의 도메인 존(zone)에 대해 마스터 네임서버인 네임서버는 다른 별개의 도메인 존(zone)에 대해서는 슬레이브 네임서버로 동작할 수 있다.

도메인 존(zone)은 마스터 네임서버에 그 원본이 존재하고 복사본을 슬레이브 네임서버의 해당 도메인 존(zone) 데이터로 복사하는 방식의 관리 메커니즘을 가진다. 이에 사용되는 메커니즘을 통칭하여 존 전송(zone transfer) 메커니즘이라 한다. 특정 도메인 존(zone)에 속하는 도메인 네임과 그 리소스 레코드에 대한 생성, 변경, 삭제는 항상 마스터 네임서버에서만 가능하다. 반면, 슬레이브 네임서버는 해당 도메인 존(zone)의 변경여부를 확인하여 변경이 발생한 경우, 마스터 네임서버로 해당 존(zone) 데이터 전송을 요청한다. 슬레이브 네임서버는 전송받은 새로운 버전의 도메인 존(zone) 데이터로 자신이 관리하고 있는 해당 도메인 존(zone) 내용을 갱신한다.

특정 도메인 존(zone)의 마스터 네임서버와 슬레이브 네임서버 구성은 도메인

존(zone)의 리소스 레코드에 일정한 데이터로 반영된다.
SOA 리소스 레코드의 <MNAME> 필드는 해당 도메인 존(zone)의 최상위 마스터 네임서버의 정보를 지정한다. <MNAME> 필드는 FQDN 형식의 도메인 네임의 값을 가진다. 이 최상위 마스터 네임서버가 해당 도메인 존(zone)의 원본 도메인 존(zone)을 가지고 있는 마스터 네임서버로써 도메인 존(zone)에 대한 수정작업이 이루어질 수 있는 서버이다. 나머지 네임서버들은 이 최상위 마스터 네임서버로부터 존 전송(zone transfer)을 받아 도메인 존(zone)을 관리하는 슬레이브 네임서버에 속하게 된다.

도메인 존(zone)의 네임서버들은 NS 리소스 레코드에 의해 지정된다.
도메인 존(zone)에 대해 NS 레코드로 지정된 DNS 서버들 중 SOA 레코드의 <MNAME> 최상위 마스터 네임서버(primary master name server)로 지정된 네임서버가 이 모든 네임서버에 대한 마스터 네임서버이다.

그림의 예에서, example.co.kr. 도메인 존(zone)은 최상위 마스터 네임서버로 ns1.example.co.kr. 네임서버를 지정하고 있다. 따라서 example.co.kr. 도메인 존(zone)은 마스터 네임서버로 ns1.example.co.kr. 서버가 그리고 슬레이브 네임서버로는 ns2.example.co.kr.과 ns3.example.co.kr. 서버를 가지고 있다. 관리자는 ns1.example.co.kr. 서버의 도메인 존(zone)을 사용하여 도메인 정보의 수정 작업을 하게 된다.

도메인 존(zone)에 지정된 NS 리소스 레코드는 리졸버에 의해서는 거의 사용되지 않는다. 상위 도메인에서 해당 도메인 존(zone)의 글루 레코드의 도움으로 네임서버 정보를 이미 파악하여 접근하기 때문이다.
이 NS 레코드 정보는 주로 도메인 존(zone)에 대한 관리 메커니즘에 사용된다.

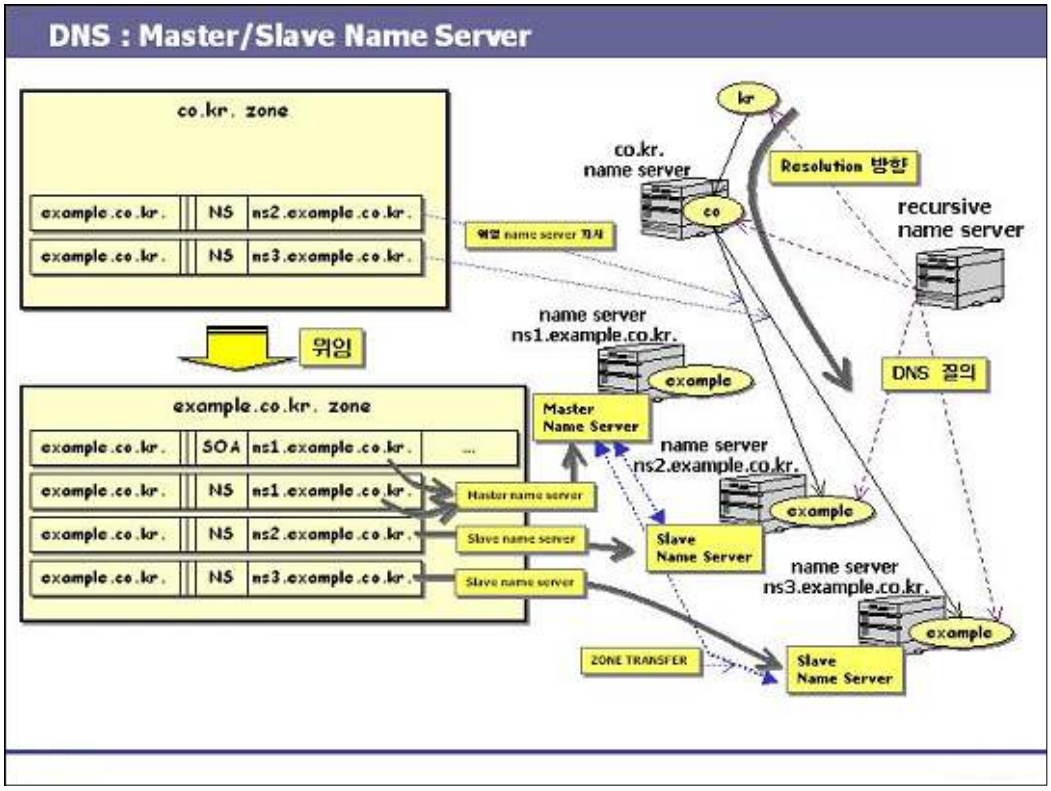
DNS NOTIFY 메커니즘은 도메인 존(zone)의 변경이 발생한 경우, 슬레이브 네임서버로 이 사실을 즉각적으로 통보해 주는 기능을 제공한다. DNS NOTIFY는 도메인 존(zone)에 대해 설정되어 있는 NS 레코드 정보를 통하여 NOTIFY를 받아야 하는 슬레이브 네임서버 리스트를 파악한다. 따라서 NS 레코드 리스트에서 누락된 슬레이브 네임서버는 DNS NOTIFY에 의한 도메인 존(zone) 변경통보에서 제외될 수 있다. 이는 해당 누락된 슬레이브 네임서버의 도메인 정보 갱신이 지연됨으로 인한 도메인 존(zone) 정보의 불일치 문제를 발생시킬 수 있다.

DNS NOTIFY는 NS 레코드들 중에서 SOA 필드의 최상위 마스터 네임서버와

일치하는 네임서버를 제외한 나머지 슬레이브 네임서버만을 대상으로 송출된다. SOA 레코드의 <MNAME> 필드, 즉 최상위 마스터 네임서버 정보는 DNS 다이나믹 업데이트(dynamic update) 메커니즘에 의해 사용된다.

DNS 다이나믹 업데이트는 먼저 생성, 삭제하고자 하는 대상 도메인 네임이 어느 도메인 존(zone)에 속해 있는지와 어느 네임서버로 접근하여 리소스 레코드의 동적갱신(dynamic update)을 요청해야 하는지를 파악하는 질의 절차를 수행한다. 이 과정에서 대상 도메인 네임이 속한 도메인 존(zone)의 SOA 레코드 정보를 얻은 후 <MNAME> 필드의 최상위 마스터 네임서버를 파악한다. 이는 리소스 레코드의 갱신은 해당 도메인 존(zone)의 마스터 네임서버에서 이루어져야 하기 때문이다. DNS 다이나믹 업데이트 메커니즘은 최상위 마스터 네임서버의 위치가 파악된 후 바로 이 마스터 서버로 직접 접근하여 DNS 동적갱신 요청을 한다. 따라서 SOA 레코드의 최상위 마스터 네임서버의 지정에 오류가 있는 경우, DNS 다이나믹 업데이트 메커니즘이 제대로 동작하지 않을 수 있다.

일반적이지는 않으나 네트워크 환경에 따라 아래와 같은 구성이 가능하다.



[그림 1-7] DNS 마스터/슬레이브 네임서버 위임구성 사례 -2

이 경우에서는 도메인 존(zone) example.co.kr.의 최상위 마스터 네임서버가 상위 도메인 co.kr.에서 NS 레코드로 지정되어 있지 않은 구성으로 되어 있다.

이 구성에서 리졸버(resolver)에 의한 통상적인 리졸루션(resolution) 절차에 있어서는 ns2.example.co.kr. 서버와 ns3.example.co.kr. 서버를 참조하게 된다. 반면, 도메인 존(zone)의 관리측면에 있어서는 ns1.example.co.kr. 서버에서 그 관리가 이루어지며 나머지 네임서버 ns2.example.co.kr.과 ns3.example.co.kr. 서버가 ns1.example.co.kr.의 마스터 존(master zone)의 내용을 복사하여 DNS 서비스를 제공한다.

흔히 최상위 마스터 네임서버가 방화벽 내부에 존재하고 외부 인터넷으로부터의 접근에 보안적 측면의 차단을 설정하는 경우에 해당한다고 할 수 있다. 최상위 마스터 네임서버를 외부에서 공격하여 마스터 존(master zone)에 대한 변경, 삭제 등을 유발하는 보안 사고로부터 보호하기 위한 목적으로 구성할 수 있다.

이 경우, DNS 다이나믹 업데이트 메커니즘은 최상위 마스터 네임서버에 직접 접근이 불가능하므로 슬레이브 네임서버로 DNS 다이나믹 업데이트 요청을 하고 해당 슬레이브 네임서버는 최상위 마스터 네임서버로 이 동적갱신 요청을 전달(forward)하는 방식으로 동작하도록 규정한다.

1.7 도메인 네임 리졸루션(Domain name Resolution)

도메인 데이터베이스가 구성된 상태에서 이 데이터베이스에서 필요한 정보를 조회하여 인터넷 응용 프로그램이 사용하기 위해서 필요한 절차가 도메인 네임에 대한 리졸루션이다. 이는 흔히 도메인 네임의 IP 주소로의 변환절차로 알려져 있다. 그러나 보다 일반적인 시각에서 볼 때, 도메인 네임이 가질 수 있는 속성정보는 단지 IP 주소만이 아니므로 도메인 네임을 IP 주소로 변환하는 용도로만 DNS 체계가 존재하는 것은 아니다. 향후 보다 다양한 형태의 도메인 네임에 대한 속성정보가 DNS의 도메인 데이터베이스에 반영되고 이를 활용한 새로운 개념의 서비스가 가능할 것으로 예상된다.

일반 단말 호스트의 입장에서 보면 도메인 네임의 리졸루션은 인터넷 상의 IP 프로토콜을 이용한 실제 통신을 함에 있어 통신대상 호스트 주소 또는 통신대상에 대한 인터넷 속성정보를 조회하기 위한 절차에 속한다. 따라서 도메인 네임에 대한 리졸루션 절차는 통신 개시절차 이전에 수행될 필요성이 있다.

일반적인 예로써 웹 브라우저 프로그램은 사용자가 입력한 URL을 근거로 접속대상 서버를 파악해야 한다. 웹 서비스의 URL은 간략하게 표현하면 http://<domain name>/<directory path>/<file name> 과 같은 형식을 갖는다. URL(Uniform Resource Locator)은 그 용어가 의미하는 바와 같이 파일과 같은 특정한 인터넷 자원(resource)에 대한 인터넷 상의 고유한 위치정보를 담고 있다.

웹 브라우저는 실제 IP 프로토콜을 사용한 통신을 개시하기 이전에 먼저 이 URL 중에서 <domain name>의 네트워크 주소를 파악하여야 한다. 이는 도메인 네임은 네트워크 주소가 아니며 네트워크 주소인 IP 주소가 있어야 IP 프로토콜을 통해 접속 대상 호스트로 IP 패킷을 전송할 수 있기 때문이다.

프로그램은 네트워크 통신을 위한 소켓(socket) 인터페이스를 호출하기 이전에 도메인 네임 리졸루션을 수행하는 함수를 호출한다. 애플리케이션 프로그램 수준에서 호출하는 네임변환(name translation) 함수로는 일반적으로 **gethostbyname()** 함수를 사용해 왔다. 이 함수는 네임 문자열을 입력값으로 받아 IPv4 주소를 되돌려 주는 기능을 제공한다. 이 과정에서 도메인 네임 리졸버 루틴을 호출하고 이를 통해 IPv4 주소 레코드에 해당하는 A 리소스 레코드에 대한 DNS 질의를 수행함으로써 도메인 데이터베이스에서 IPv4 주소를 조회한 후 그 결과를 되돌려 준다.

웹 브라우저 프로그램은 응답된 IPv4 주소를 착신 네트워크 주소로 설정하여 소켓(socket) 인터페이스 함수를 호출하고 통신절차를 개시한다.

따라서 일반적인 인터넷 통신 응용 프로그램들은 사용자가 입력하는 도메인 네임에 대한 속성정보의 조회절차를 거친 후 본격적인 통신절차를 개시하는 구조를 갖는다.

단말 호스트의 도메인 네임 리졸루션 기능을 제공하는 리졸버는 스터브 리졸버이다. 스터브 리졸버는 리커시브 네임서버의 IP 주소를 환경 설정 데이터로 가지고 있으면서 이 리커시브 네임서버로 DNS 질의 메시지를 송출하고 최종 응답을 기다린다.

리커시브 네임서버는 단말 클라이언트로부터 요청되는 리커시브 질의 요청에 대해 네임서버에 함께 포함하여 구현된 리졸버 루틴으로 이 요청을 넘겨 리졸버 루틴이 이를 처리하도록 한다. 리졸버 루틴은 캐시나 네임서버의 공유 데이터베이스 등의 로컬 정보를 조회하여 해당 요청 도메인 네임에 대한 정보가 있는 경우, 바로 이를 응답으로 리턴한다. 만일 그 해당 정보가 부재한 경우, 리졸버는 루트 네임서버로부터 시작하는 도메인 트리 구조를 순차적으로 조회하기 시작한다. 이를 위해 리졸버는 리커시브 네임서버의 환경 설정 데이터로 존재하는 루트 네임서버의 IP 주소 정보를 참조한다. 리졸버는 도메인 트리 구조를 순차적으로 탐색하면서 원하는 도메인 데이터베이스 영역(zone)이 존재하는 위치, 즉 네임서버의 IP 주소를 파악하고 해당 네임서버로 DNS 질의를 수행한다. 해당 네임서버로부터의 응답 메시지를 단말 클라이언트 호스트로 DNS 응답 메시지를 사용하여 응답한다. 이와 함께 리졸버 루틴은 응답결과를 로컬 캐시에 저장한다.

모든 네임서버가 리커시브 네임서버로 동작하지는 않는다. 일반적으로 호스트 단말에 설정하는 DNS 네임서버는 리커시브 네임서버이다. 이는 주로 ISP 등에 의해 운영되고 사용자에게 그 IP 주소를 공개한다. (예, KT의 168.126.63.1)

대부분의 네임서버는 리커시브가 아닌 모드(non-recursive mode)로 동작하는 기본적인 네임서버이다. 이를 이터레이티브 네임서버(iterative name server)라고도 한다. 리커시브 모드로 동작하지 않는 네임서버를 편의상 네임서버(name server)라 하고 리커시브 모드로 동작하는 네임서버를 리커시브 네임서버(recursive name server)라고 구분하여 사용하기로 한다.

네임서버는 자신이 보유한 도메인 데이터베이스 영역(zone)에 대해서만 DNS

응답을 한다. 해당 도메인 데이터베이스 영역을 벗어난 영역의 도메인 네임에 대한 질의에 대해서는 현재 보유한 도메인 영역으로부터 가장 근접한 위임영역의 네임서버 정보를 참조정보로 응답한다. 즉, 리졸버가 다음 단계의 네임서버로 탐색을 계속할 수 있도록 다음 네임서버 정보를 제공하는 역할만 한다.

리커시브 네임서버가 클라이언트를 대신해 리졸버의 전 기능을 수행함에 비해 네임서버(name server)는 권한(authority)을 갖는 도메인 데이터베이스 정보에 대한 응답만 제공하며 리졸버 기능을 사용하지 않는다. 따라서 이와 같이 리커시브가 아닌 모드(non-recursive mode)로 동작하는 네임서버(name server)를 단말 호스트에 DNS 서버 또는 네임서버로 지정하는 경우, 호스트는 도메인 네임에 대한 리졸루션 기능을 제대로 수행할 수 없게 된다.

1.8 관련 표준 목록

DNS는 오랫동안 인터넷 상의 성공적인 네임체계로 자리해 왔다. 전 세계에 헤아릴 수 없이 많은 네임서버가 구축되어 있고 모든 호스트는 리졸버 루틴을 통해 인터넷 상에 존재하는 도메인 데이터베이스에 접근하여 필요한 정보를 조회하고 있다.

이 장에서는 이 지침서에서 미처 다 다룰 수 없는 상세한 DNS 프로토콜 및 구조에 대한 심층적인 파악과 참조가 가능하도록 관련 IETF 표준 목록을 정리하였다. DNS 관련 전체 RFC 문서는 상당한 수에 이른다. 여기에서는 일반적으로 사용되지 않는 기능의 확장을 정의한 문서들을 제외한 문서를 중심으로 추려서 정리하였다. 이에는 향후 사용이 일반화될 수 있는 기능에 대한 문서들과 DNS 운영상의 참고 문서와 DNS 개념 명확화를 위한 문서들을 포함하였다. 단, IPv6 관련 확장 표준 문서는 여기에서 제외하였다.

RFC 번호	제목,	비고
RFC1033	DOMAIN ADMINISTRATORS OPERATIONS GUIDE	DNS 관리자의 운영가이드
RFC1034 (STD13)	DOMAIN NAMES - CONCEPTS AND FACILITIES	개선 보완된 DNS 체계 기본 표준문서
RFC1035 (STD13)	DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION	개선 보완된 DNS 구현 기본 표준문서
RFC1912	Common DNS Operational and Configuration Errors	DNS 운영상의 구성 오류 사례
RFC1982	Serial Number Arithmetic	SOA 레코드 serial 번호의 산술계산
RFC1995	Incremental Zone Transfer in DNS	IXFR 신규 정의
RFC1996	A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)	Zone의 갱신에 대한 NOTIFY
RFC2136	Dynamic Updates in the Domain Name System (DNS UPDATE)	도메인 네임에 대한 동적 갱신
RFC2181	Clarifications to the DNS Specification	DNS에 대한 개념 명확화
RFC2308	Negative Caching of DNS Queries (DNS NCACHE)	실패한 DNS 질의에 대한 캐시 처리
RFC3425	Obsoleting IQUERY	DNS 질의 코드 IQUERY 폐기
RFC2535	Domain Name System Security Extensions	DNS에 대한 보안 확장, 관련 RR 정의
RFC2845	Secret Key Transaction Authentication for DNS (TSIG)	
RFC2930	Secret Key Establishment for DNS (TKEY RR)	
RFC2931	DNS Request and Transaction Signatures (SIG(0)s)	
RFC3007	Secure Domain Name System (DNS) Dynamic Update	
RFC3008	Domain Name System Security (DNSSEC) Signing Authority	
RFC3090	DNS Security Extension Clarification on Zone Status	
RFC3226	DNSSEC and IPv6 A6 aware server/resolver message size requirements	
RFC3445	Limiting the Scope of the KEY Resource Record (RR)	

RFC 번호	제목,	비고
RFC1886	DNS Extensions to support IP version 6	DNS의 IPv6 확장 RR 및 도메인 정의 AAAA RR/IP6.ARPA 사용 귀결 진행
RFC2874	DNS Extensions to Support IPv6 Address Aggregation and Renumbering	
RFC3364	Tradeoffs in Domain Name System (DNS) Support for Internet Protocol version 6 (IPv6)	
RFC3226	Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)	
RFC3152	Delegation of IP6.ARPA	

[표 1-3] DNS 관련 전체 RFC 문서

2장 BIND

2.1 BIND 소개

BIND(Berkeley Internet Name Domain)는 현재 전 세계에서 가장 많이 사용되는 DNS용 응용프로그램이다. 이 프로그램은 유닉스, 리눅스, 윈도우 등 대부분의 운영체제에서 사용이 가능하며 도메인 네임서버와 지원 라이브러리 그리고 클라이언트 프로그램이 함께 배포된다.

BIND에 대한 정보는 http://www.isc.org/sw/bind에서 얻을 수 있으며 메일링 리스트를 통한 공개된 논의가 진행되고 있다.

□ BIND 8.x 소개

BIND 8은 기존의 BIND 4.x를 개선한 것으로 새로 포함된 주요 기능은 다음과 같다.

- o DNS Dynamic Update
- o DNS Change Notification
- o Completely new configuration syntax
- o Flexible, categorized logging system
- o IP-address-based access control for queries, zone transfer and updates that may be specified on a zone-by-zone basis
- o More efficient zone transfers
- o Improved performance for servers with thousands of zones
- o The server no longer forks for outbound zone transfer
- o Many bug fixed

□ BIND 9.x

BIND 9는 기존의 BIND 구조를 전반적으로 다시 작성한 버전으로 BIND 9에서 포함된 주요 기능은 다음과 같다.

- o DNS 보안
  - DNSSEC(signed zones)
  - TSIG(signed DNS request)
- o IPv6

- IPv6 소켓을 통한 DNS 질의에 대한 응답
- IPv6 관련 리소스 레코드(A6, DNAME, etc.)
- 비트스트링 레이블(Bitstring Labels) - 폐기됨
- 실험적인 IPv6 리졸버 라이브러리

o DNS Protocol Enhancement
- IXFR
- DDNS
- Notify
- EDNSO

o BIND 8.x 보다 표준안에 부합

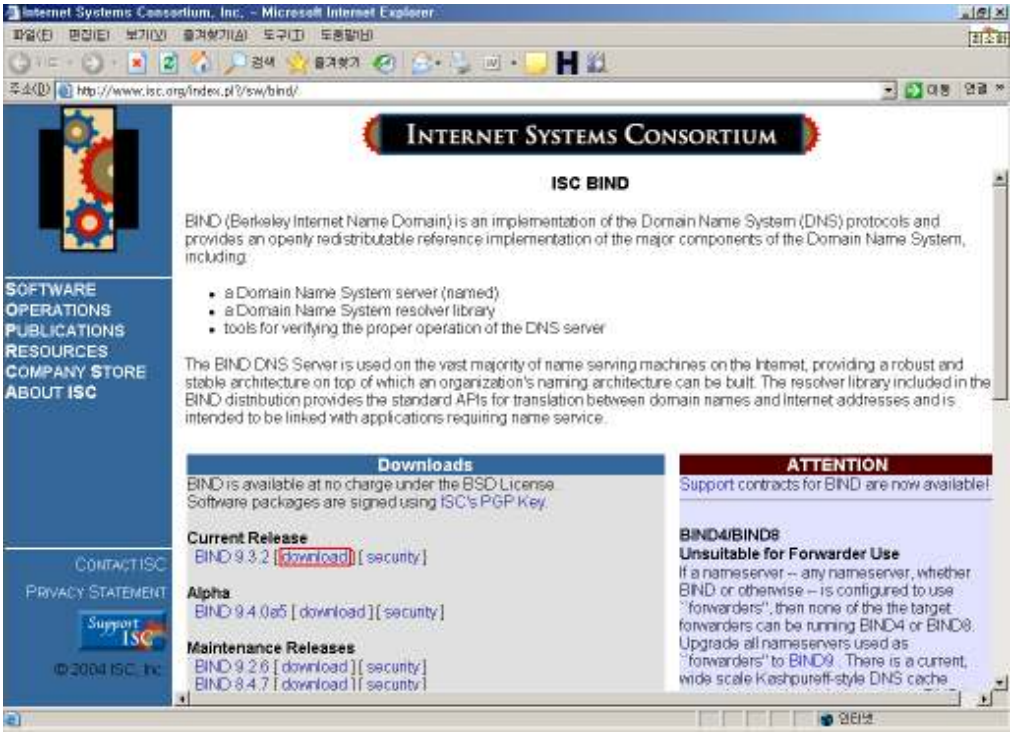
o View
- 하나의 서버가 여러 뷰를 지원할 수 있다. 다시 말해서 특정 클라이언트에게는 내부 뷰를 보여주고 , 다른 클라이언트에게는 외부 뷰만을 보이도록 설정할 수 있다.

o 다중 프로세서 지원

o BIND 8.x 보다 이식이 용이한 구조

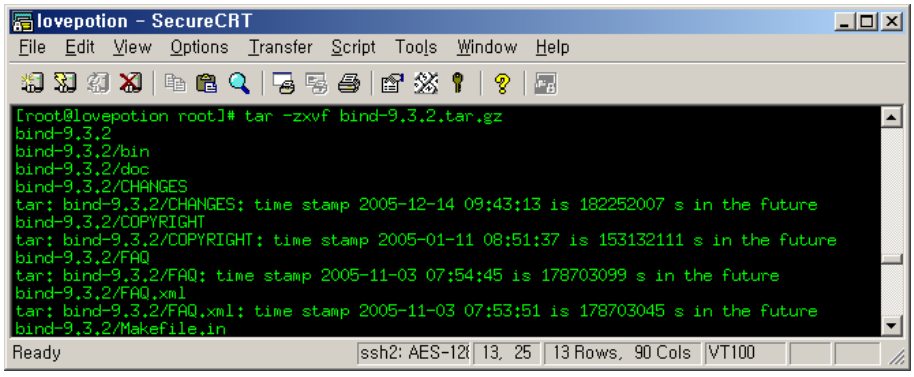
2.2 BIND 설치하기

BIND는 http://www.isc.org/sw/bind에서 다운받을 수 있다.



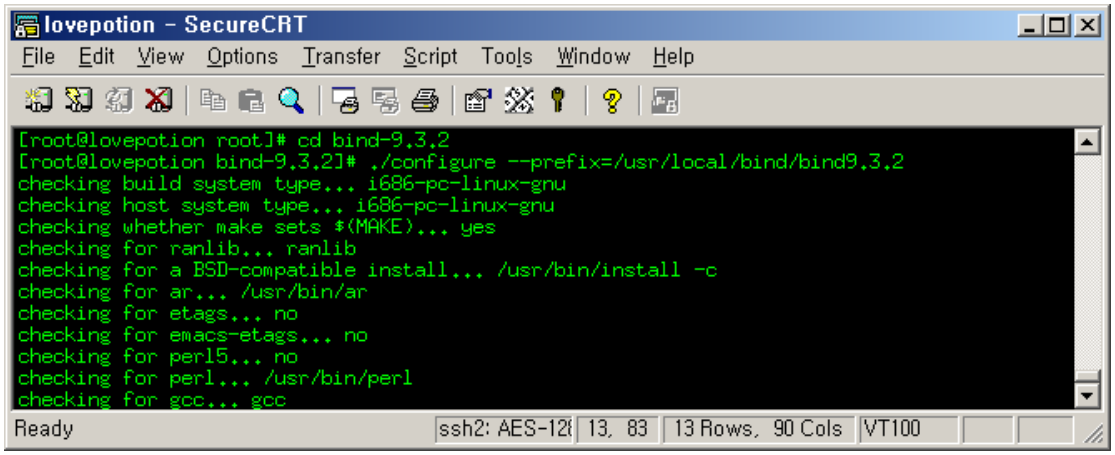
[그림 2-1] ISC BIND 화면

BIND설치과정은 압축된 파일을 해제한 후, 환경을 설정(./configure)하고 make를 수행하면 된다. 다음 그림은 tar 명령을 이용하여 다운받은 파일을 압축해제하는 화면이다. 참고로, BIND8의 설치는 src/port/ 아래 vendor OS별 Makefile.set을 수정하여 make depend; make; make install 로 설치하고 있고, BIND9 설치 시에만 GNU configure and build systems 도입(configure; make; make install)하여 아래와 같은 설치과정을 거친다.



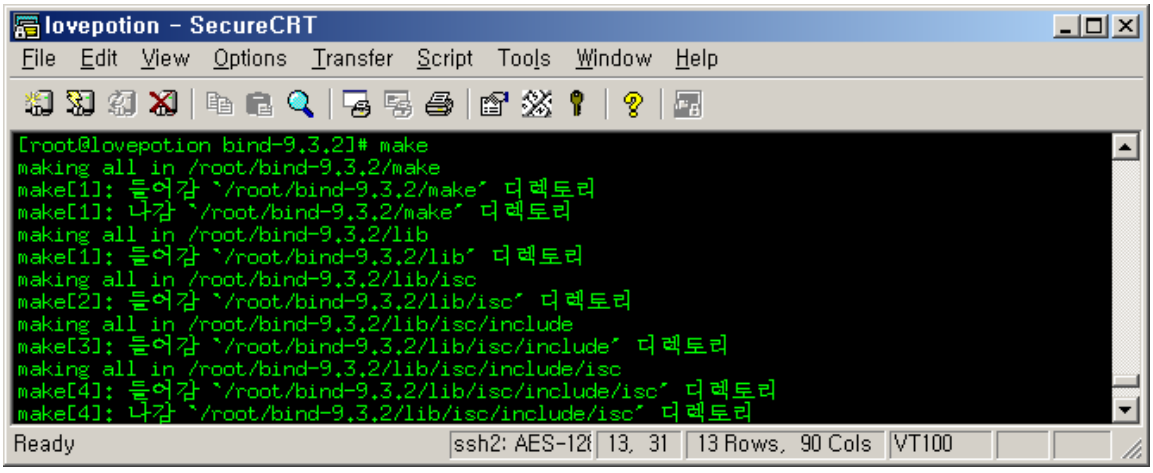
[그림 2-2] 압축해제 화면

압축을 해제 후 ./configure 명령을 수행한다. 이 과정은 현재 시스템의 상태를 확인하고 관련된 파일들의 존재 유무, 사용할 옵션 등을 확인하는 과정이다. prefix 옵션을 이용하여 자신이 원하는 설치경로를 정할 수 있다.



[그림 2-3] configure 명령 수행 화면

이 과정이 끝나면 make를 이용하여 BIND를 설치한다.
make가 끝나면 root 권한으로 make install을 수행하면 설치과정이 마무리된다.



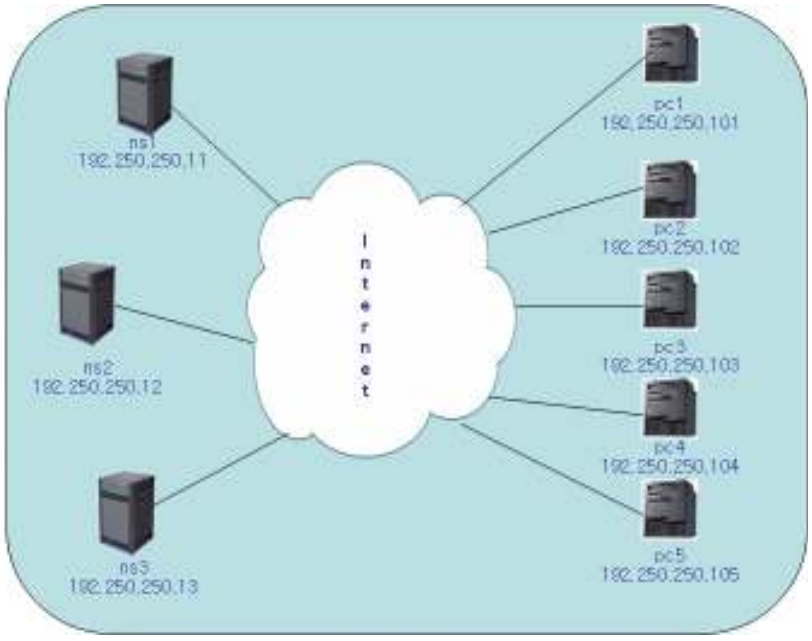
[그림 2-4] make 수행 화면

2.3 BIND 설정하기

BIND를 설치한 후 해야 할 작업은 영역데이터(zone) 파일들을 설정하는 것이다. 우선 영역 데이터 파일 중에는 호스트 이름을 주소로 매핑하는 파일인 db.DOMAIN과 반대로 주소를 이름으로 매핑하는 db.ADDR 파일이 있다.

이들 영역 파일들을 하나로 묶어주는 환경 설정 파일이 필요한데, BIND 8이나 BIND 9에서는 name.conf라는 이름을 가지고 있다. 앞서 소개한 영역데이터 파일의 형식은 표준으로 정해진 것이므로 어떠한 DNS 소프트웨어에서나 동일하지만, 환경 설정 파일의 문법은 소프트웨어마다 다를 수 있다.

다음 그림을 예로 들면서 영역파일을 설정하는 과정을 보이도록 하겠다. example.co.kr도메인을 사용하는 회사의 망주소는 192.250.250/24이고 이 망에는 네임서버가 2대 운영되고 있다. 웹서버와 메일서비스는 하나의 서버에서 제공하는 것으로 가정하고 5대의 PC가 운영되고 있다고 가정한다.



[그림 2-5] example.co.kr. 도메인 구성

□ 영역 데이터 파일

DNS 검색 작업은 대소문자를 가리지 않기 때문에 영역 파일에 이름을 입력할 때 대문자나 소문자, 또는 둘 다 섞어 사용해도 된다. 그러나 검색과정에서는

대소문자를 구별하지는 않지만 대소문자 자체는 보존된다. 예를 들어 영역데이터에 Abc.example.co.kr이라는 레코드를 추가하면, abc.example.co.kr이라는 이름으로 검색을 해도 그 레코드를 찾을 수 있지만 결과에 대문자 A가 그대로 나타난다.

리소스 레코드는 반드시 첫 번째 열부터 시작해야 한다. DNS RFC 문서에 나타난 예제의 리소스 레코들 간에 순서가 있고 대부분의 사람들은 이러한 순서를 따르는데, 이 순서를 반드시 따를 필요는 없다. 일반적인 영역 데이터 파일 내의 리소스 레코드들의 순서는 다음과 같다.

SOA 레코드
  본 영역에 대한 권한(authority)을 나타낸다.
NS 레코드
  본 영역에 대한 네임서버를 나열한다.
가타 레코드
  본 영역 내의 호스트에 대한 데이터이다.

[표 2-1] 일반적인 RR 순서

기타 레코드 중에는 이름을 주소로 매핑하는 A RR과 반대로 주소를 이름으로 매핑하는 PTR RR, 전형적인 이름을 위한 CNAME RR이 있으며, IPv6 주소 매핑을 위한 AAAA RR도 여기에 해당한다. 또한 축약어를 이용할 수도 있다.

주석문이나 공백 라인이 있으면 영역 데이터 파일을 읽기가 수월한데, 영역데이터 파일에 있는 주석문은 세미콜론(;)으로 시작하며 세미콜론부터 그 라인의 끝까지 모두 주석문이다.

o 영역의 기본 TTL(Time To Live) 설정

TTL 값을 설정하기 앞서 먼저 BIND 버전을 확인할 필요가 있다. 그 이유는 TTL 기본값이 BIND 8.2를 기준으로 바뀌었기 때문이다. BIND 8.2보다 이전 버전에서는 SOA 레코드의 마지막 필드가 TTL 기본값이 된다. 그러나 BIND 8.2가 발표되기 바로 전에 RFC 2308이 발표되었고, 여기서는 SOA 레코드의 마지막 필드의 의미를 부정적 캐싱의 TTL로 정의하였다. 부정적 캐싱의 TTL 값이란 해당 영역에 대한 부정적 응답(특정 도메인 네임이 존재하지 않거나 또는 찾고자 하는 데이터 종류가 해당 도메인 네임에 없음을 의미하는 대답)을 얼마나 오랫동안 원격 네임서버가 캐싱하고 있어야 하는가를 정의하는 값이다.

BIND 8.2 및 이후 버전에서는 영역의 기본 TTL 값을 $TTL 제어 구문을 이용해서 한다. $TTL 구문이 선언되면 이후의 레코드는 그 생존 시간을 따른다.

네임서버는 질의에 대한 응답 속에 이 TTL 값을 넣어서 반환하며, 응답을 받은 서버는 응답데이터를 그 TTL 시간동안 캐싱한다. 데이터가 자주 바뀌는 것이 아니라면 TTL을 며칠 이상의 큰 값으로 지정할 수 있다. 가장 길면서도 합당한 값은 일주일 정도이며, 짧게는 1시간 정도로 지정할 수 있다. 그보다 작은 TTL을 이용하는 것은 DNS 트래픽이 많이 유발되기 때문에 바람하지 않다. 일반적으로는 3시간 정도를 TTL 값으로 이용한다. BIND 8.2보다 이전 버전의 네임서버를 운영한다면 $TTL 구문을 추가하면 안 된다. 이 경우 네임서버가 이 구문을 이해하지 못하고 문법 오류로 취급한다.

o SOA(Start Of Authority) 레코드

SOA RR은 네임서버가 영역의 데이터에 대한 가장 최고의 정보 제공 처임을 나타낸다. 네임서버는 다음의 SOA 레코드 때문에 example.co.kr 영역에 대한 권한을 갖는다. 모든 db.DOMAIN 및 db.ADDR 파일마다 SOA 레코드가 필요하며, 하나의 영역 데이터 파일에는 반드시 하나의 SOA 레코드만 있어야 한다.

```
example.co.kr.   IN   SOA   ns1.example.co.kr.   root.example.co.kr.  (
                                2006050100      ;시리얼 번호
                                3h              ;3시간 후 리플래시
                                1h              ;1시간 후 재시도
                                1w              ;1주일 후 만료
                                1h)             ;1시간의 부정적 캐싱 TTL
```

[표 2-2] SOA RR 설정 예

도메인이름 example.co.kr.은 반드시 파일의 첫 열에서 시작해야하며, 도메인 이름의 끝은 반드시 점(.)으로 끝나야 한다. 이 점(.)을 생략하는 경우 도메인 이름이 자동으로 추가되어 엉뚱한 결과를 초래한다.

IN은 인터넷을 의미한다. IN 클래스 이외에도 다른 클래스들이 있지만 거의 사용되지 않는다. 클래스 필드는 선택적이어서 이를 생략하면 환경설정 파일의 구문으로부터 클래스를 결정한다.

SOA 다음에 오는 이름 ns1.example.co.kr.은 이 데이터에 대한 주 마스터 네임서버의 이름이다. 그 다음에 오는 root.example.co.kr.은 제일 처음의 '.'을 '@'로 바꾸면 해당 영역에 대한 책임자의 이메일 주소가 된다. 종종 메일주소로서 postmaster, hostmaster, admin 등을 볼 수 있다. 네임서버는 SOA에 적힌 이런 메일 주소를 이용하지 않는다. 다만 영역에 문제가 생겼을 때 명시된 주소로 이 메일메시지를 보낼 수 있도록 사람이 보기위한 용도로 이용된다.

SOA 레코드는 괄호가 시작되는 곳에서부터 괄호가 끝나는 지점까지 여러 라인을 차지할 수도 있다. 괄호 속에 있는 필드들의 대부분은 슬레이브 네임서버에 의해 이용되는데, [표 2-2]를 간략히 설명하면 시리얼번호를 기준으로 데이터의 갱신 여부를 판단하게 되며, 정상적인 경우 3시간마다 확인을 하도록 되어 있고, 이 때 장애가 있는 경우 1시간 후 다시 재시도를 하고, 이렇게 얻어진 데이터 파일은 1주일간 유효하며, 부정적 정보(특정도메인 네임이 존재하지 않거나 또는 찾고자 하는 데이터 종류가 해당 도메인 네임에 없음을 의미)를 얻을 경우 그것이 잘못되었다는 사실을 1시간동안 유지함을 뜻한다.

o NS(Name Server) 레코드

이것은 영역에 대한 각 네임서버를 나타내며, 각 네임서버당 하나의 NS를 추가한다. 따라서 복수개의 네임서버가 운영되는 경우 여러 줄의 NS 레코드가 나올 수 있다.

□ db.DOMAIN 파일

o A(Address) 레코드와 CNAME(Canonical Name) 레코드

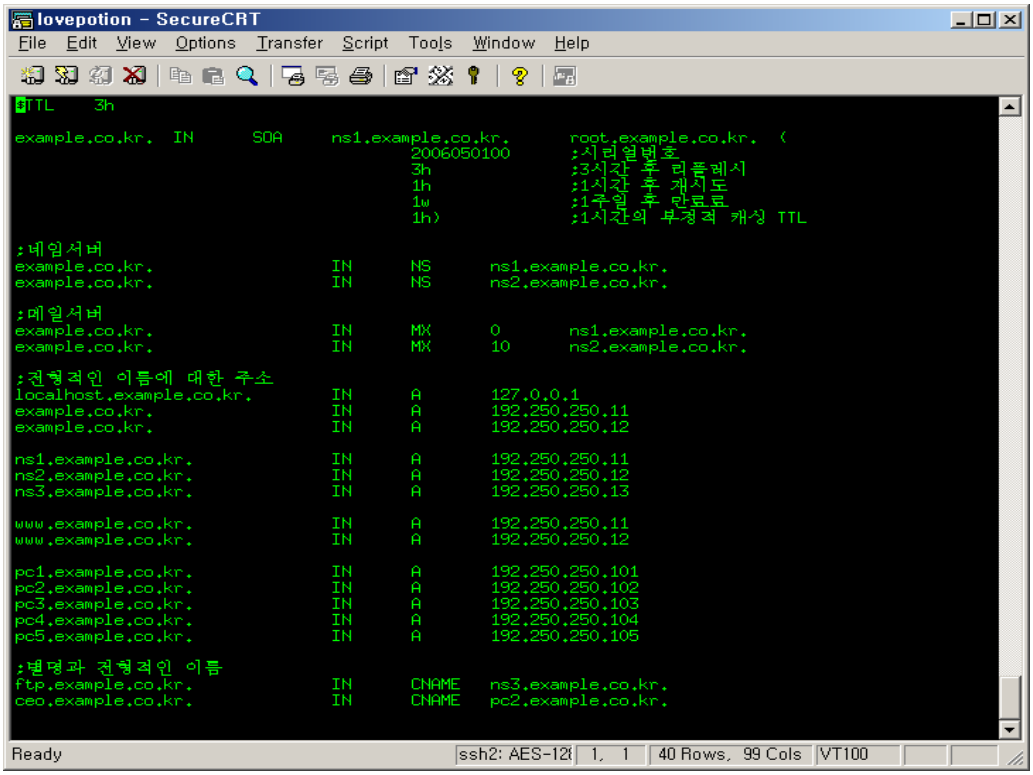
A 리소스 레코드는 호스트 네임을 주소로 맵핑하는 레코드로, A는 주소(Address)를 나타낸다. 이름은 하나지만 주소가 두개 있는 호스트의 경우 주소 레코드를 두개 가지고 있을 수 있다. 호스트 테이블 검색하는 것과는 달리, DNS 탐색은 하나의 이름에 대하여 하나 이상의 주소를 반환할 수 있다. 따라서 주소를 둘 이상 가지고 있는 호스트에 대한 질의에는 둘 이상이 반환된다. 일반적으로 망에서 가까운 주소를 앞에 적고, 멀리 있는 주소를 뒤에 적은 응답을 할 것이다. 이러한 기능은 주소 정렬(address sorting)이라고 한다. 주소 정렬이 적용되는 경우가 아니라면 질의마다 대답이 순환되어 반환되므로 순서가 다르게 나열된다. 이러한 라운드 로빈(round-robin) 기능은 BIND 4.9에서 처음 소개되었다.

CNAME(Canonical Name) 리소스 레코드는 별명을 그에 해당하는 전형적인 네임으로 맵핑한다. 네임서버가 CNAME 레코드를 다루는 방법은 호스트 테이블에서 별명이 다루어지는 방법과는 다르다. 네임서버가 어느 이름을 찾을 때 CNAME 레코드를 발견하면 그 이름을 전형적인 네임으로 바꾸어서 다시 그 새로운 이름을 탐색한다. 일반적으로 다중 네크워크를 갖는 호스트가 있다면 특정 주소에 고유한 이름의 A 레코드를 생성한다. 이때 일반 사용자에게는 CNAME 레코드에 나타난 이름만을 드러내고 시스템 관리목적으로 이용하는 A 레코드에 나타난 이름은 숨기는 것이 일반적이다.

'모든 경우에서 CNAME 레코드 대신에 A 레코드를 이용해도 좋은가?' 하는 의문이 있을 수 있는데, 대부분의 애플리케이션들에서는 문제가 발생하지 않는다. 왜냐하면 애플리케이션들은 IP 주소를 찾았는지에 대해서만 중요하게 생각하기 때문이다. 그러나 sendmail에서는 다르다. 일반적으로 sendmail은 메일 헤더의 별명들을 전형적인 네임으로 대치한다. 이런 전형적인 네임으로 바꾸기(Canonicalization)는 오직 메일 헤더 내의 이름이 관련 CNAME 데이터를 가지고 있을 때만 발생한다. 만일 CNAME 레코드를 이용하지 않으면 메일 서버 호스트에 대한 가능한 별명을 sendmail 환경 설정을 잡아줄 필요가 있다. 이 sendmail의 문제 이외에 사용자들이 자신들의 .rhosts 파일에 입력한 전형적인 네임을 확인하려고 할 때도 혼동이 발생한다. 주소 데이터를 가지고 있는 이름은 그렇지 않지만, CNAME 데이터를 가지고 있는 IP주소를 탐색하여 전형적인 네임을 얻어야 한다.

o MX(Mail Exchange) 레코드

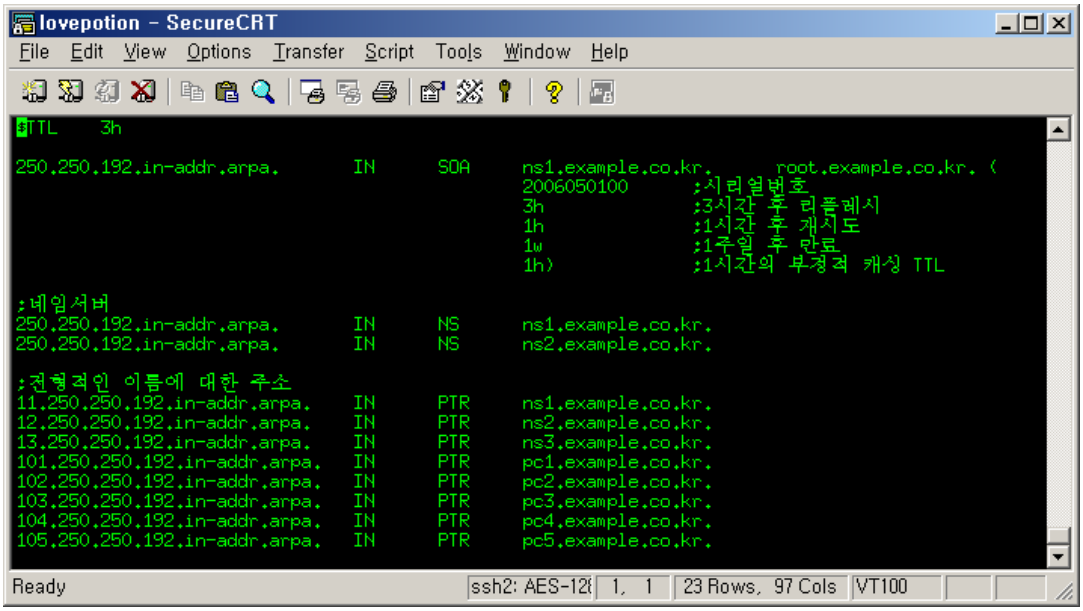
DNS를 이용하면 기존의 HOST.TXT(/etc/hosts)를 이용하는 것보다 보다 진보된 메일 라우팅이 가능하다. HOST.TXT만을 이용하는 경우 테이블에서 알아낸 IP 주소로 메일을 전송하여 실패한 경우 다시 전송하거나 송신자에게 되돌려준다. 이에 반해 DNS를 이용하면 백업 호스트를 지정할 수 있다. 다시 말하면 처음 메일서버에서 장애가 발생한 경우 다른 서버에게 대신 전송할 수 있도록 해준다. 이러한 MX 레코드에 대한 설명은 1장을 참조하면 된다.



[그림2-6] db.example.co.kr 파일

o db.ADDR 파일

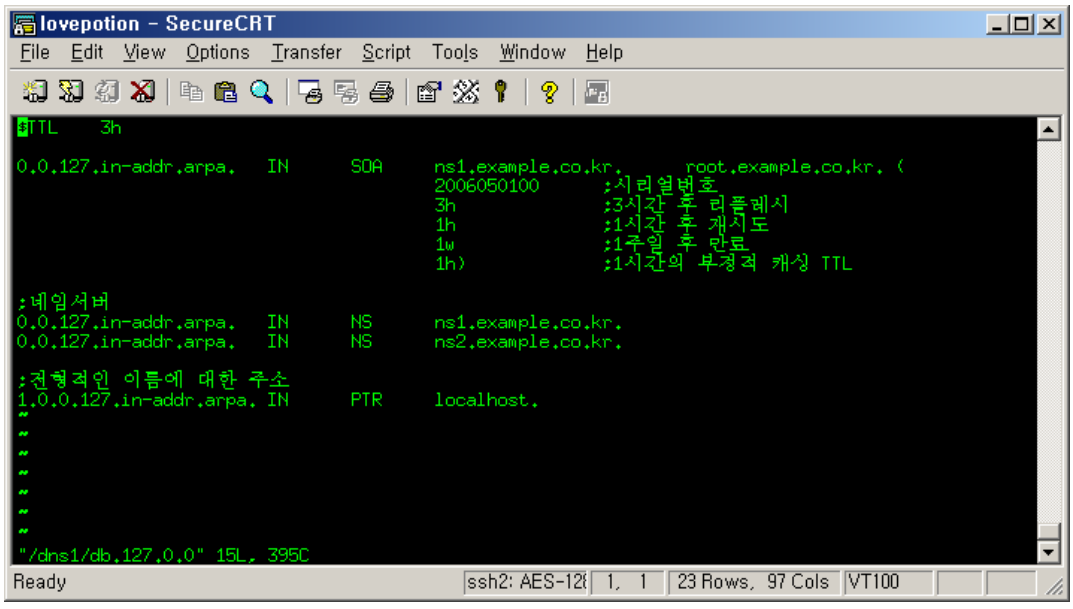
db.ADDR 파일은 주소를 이름으로 매핑하는 기능을 제공한다. 예를 들면 example.co.kr. 도메인의 경우 주소를 이름으로 매핑하는 파일을 db.192.250.250 파일이 될 수 있다. 여기에는 PTR 레코드들이 추가되어 전형적인 이름을 주소로 매핑할 수 있게 한다. 다음 그림은 db.192.250.250 파일을 보여준다.



[그림2-7] db.192.250.250 파일

o db.127.0.0 파일

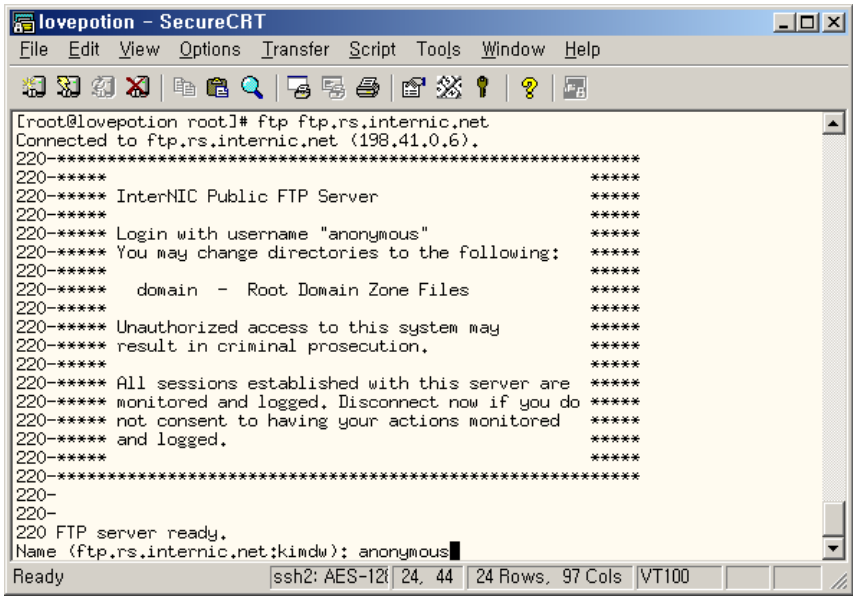
루프백 네트워크를 처리하기 위한 파일로서 대부분의 호스트가 로프백을 이용하지만 127.0.0 망을 담당하는 서버가 없으므로 반드시 설정하는 것이 좋다.



[그림 2-8] db.127.0.0 파일

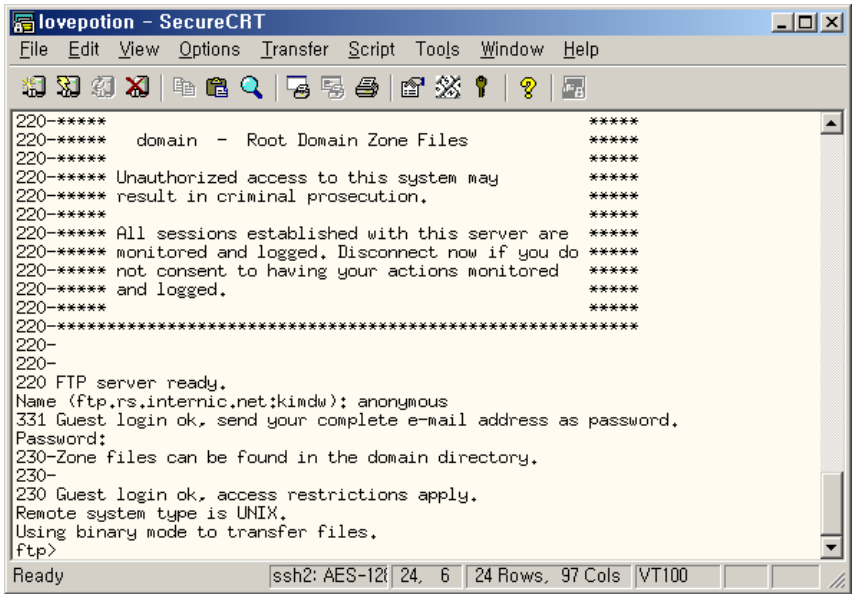
o db.cache 파일

루트 도메인에 대한 네임서버들의 위치를 담고 있는 파일로서 직접 작성하는 것이 아니라 ftp.rs.internic.net에서 가져올 수 있다. 다음 그림들은 Internic의 ftp 서버에 접속하여 named.root 파일을 가져오는 과정을 보여준다. [그림 2-9]는 ftp 명령으로 ftp.rs.internic.net에 처음 접속했을 때 나타나는 화면이다.



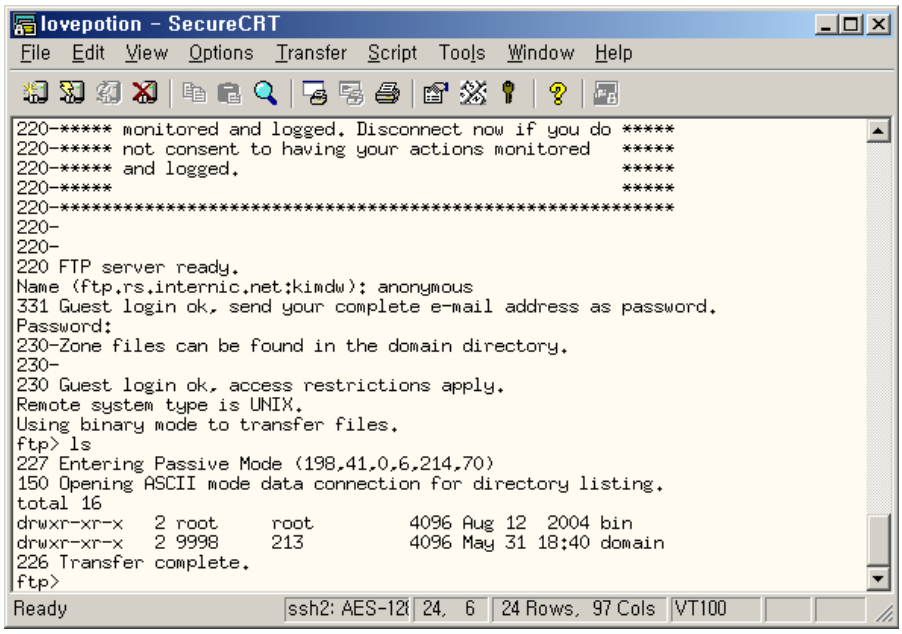
[그림 2-9] ftp ftp.rs.internic.net 실행화면

이름에 anonymous(익명)라고 입력하고 암호로 접속한 사람의 이메일 주소를 입력하면 다음 화면이 나타난다.



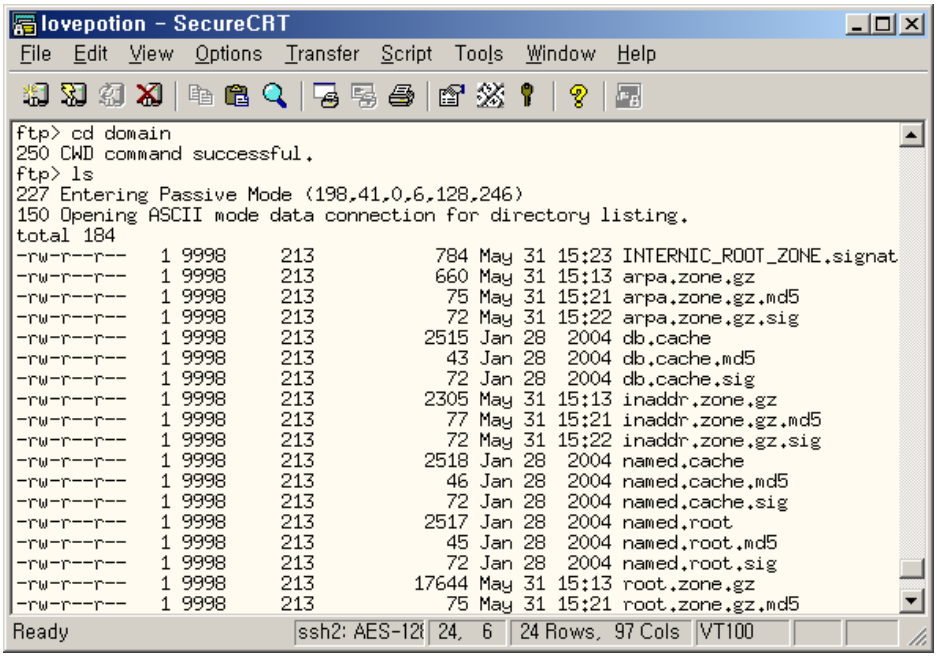
[그림 2-10] 접속 후 초기화면

접속 후 존파일들이 domain 디렉토리에 있다는 안내를 볼 수 있다. ls나 dir 명령을 통해 현재 폴더의 내용을 확인할 수 있다.



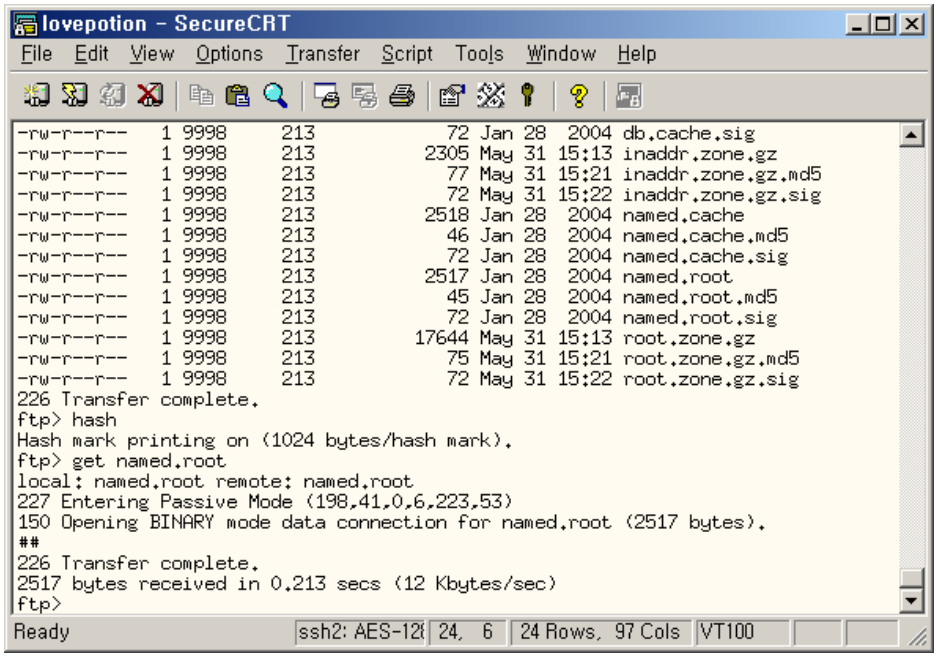
[그림 2-11] ls 명령으로 현재 폴더 내용을 확인

cd 명령으로 domain 디렉토리로 이동하여 db.cache나 named.root를 다운로드하면 된다.



[그림 2-12] cd 명령어로 domain 폴더로 이동하여 파일 확인

get 명령으로 named.root 다운받을 수 있다.



[그림 2-13] get 명령으로 파일을 다운로드

□ BIND 환경 설정 파일

BIND에서는 환경설정 파일(configuration file)을 통하여 서버가 영역데이터 파일들을 참조하게 된다. 영역 데이터 파일들은 DNS 표준을 따르지만 BIND의 환경설정 파일은 BIND만의 고유한 형태이다.

BIND 환경설정 파일의 문법은 BIND 4와 BIND 8 사이에서 매우 큰 변화가 있었다. BIND 8과 BIND 9은 동일하다. 이미 BIND 4의 환경 설정 파일을 가지고 있다면 named-bootconf라는 프로그램을 이용하여 BIND 8이나 9의 환경설정 파일로 변환할 수 있다. 이 파일의 위치는 BIND 8과 BIND 9가 다른데, BIND 8에서는 src/bin/named-bootconf이고, BIND 9에서는 contrib /named-bootconf이다.

o BIND 8, 9의 환경 설정파일

BIND 8과 9의 환경설정 파일의 주석문은 다음에 나오는 C 스타일, C++스타일, 쉘(shell) 스타일을 이용해야 한다. BIND 8부터는 ; 가 설정구문의 끝을 나타낸다.

```
/* C 스타일 주석문 */
// C++ 스타일 주석문
# 쉘 스타일 주석문
```

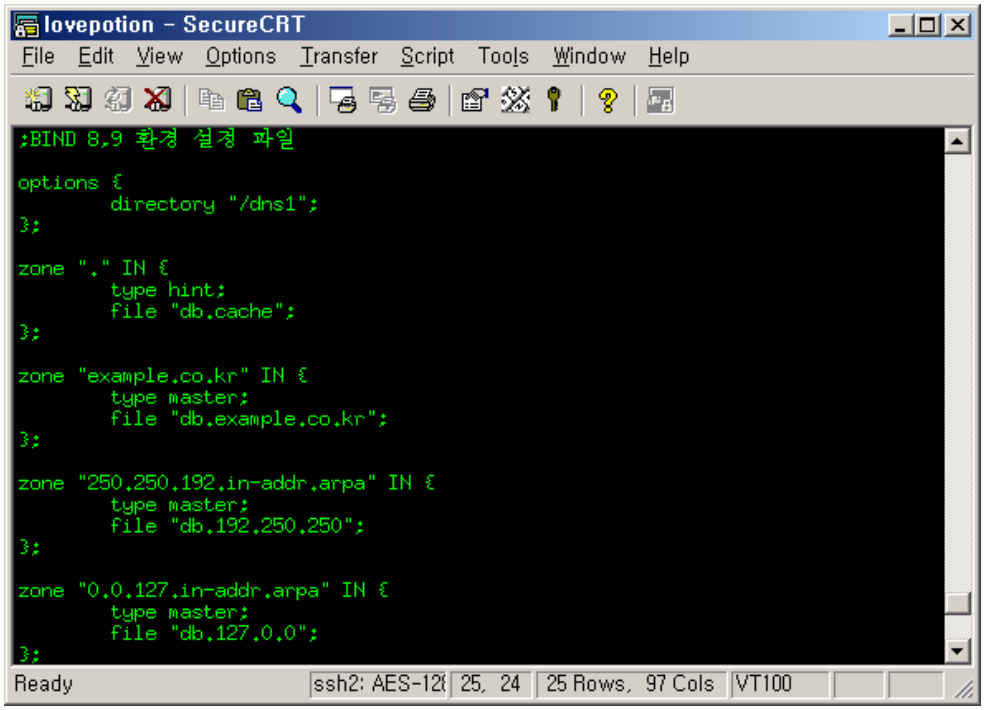
[표 2-3] BIND 8, 9의 주석문

또한 파일 위치 지정도 옵션 중 하나로 들어갔고, 이외에도 다른 옵션들이 있을 수 있다. 다음은 BIND 8과 9의 파일 위치 지정 예이다.

```
options {
        directory "/dns1";
};
```

[표 2-4] BIND 8, 9의 파일 위치 지정

BIND 8과 9에서 읽어야 하는 파일을 지정하기 위해 zone이라는 키워드로 시작하고 도메인이름과 클래스 (IN)이 뒤따라 나온다. 중괄호 안에서 type 값은 주 도메인 서버인 경우 master로, 보조 도메인 서버인 경우 slave로 설정하고, 다음에 파일이름이 나온다. 다음은 앞에 나온 example.co.kr. 도메인의 /etc/named.conf 예이다. 여기서 db.cache의 type이 hint인 점을 주목할 필요가 있다.



[그림 2-14] BIND 8, 9의 named.conf 예

□ 축약

BIND의 경우 축약을 이용할 수 있는데, db.DOMAIN 파일에서 주로 이용하는 축약 방법에는 도메인 네임 자동 확장과 @ 기호, 그리고 마지막 이름 반복 기능이 있다.

o 도메인 네임 자동 확장

환경설정 파일에서 zone 라인의 둘째 필드는 도메인이름을 나타낸다. 이 도메인이름은 축약어를 이용할 수 있다. 점(.)으로 끝나지 않는 영역 데이터 파일 내의 모든 이름들을 바로 이 기원을 뒷부분에 추가한다. 물론 각각의 영역데이터 파일들은 기원이 서로 다르다. 따라서 다음과 같이 축약이 가능하다. 여기서 ns2 다음에 (.)이 없으면 자동으로 도메인 이름이 확장되어 ns2.example.co.kr과 같이 된다. 완전한 이름을 입력하면서 마지막 (.)을 생략하는 경우 자동 확장이 일어나서 엉뚱한 결과를 초래할 수도 있다.

```
ns1.example.co.kr.      IN      A       192.250.250.11      ;완전한 이름사용
ns2                     IN      A       192.250.250.12      ;도메인 자동 확장 이용
```

[표 2-5] db.example.co.kr에서 도메인 자동 확장 예

```
11.250.250.192.in-addr.arpa.    IN  PTR    ns1.example.co.kr.    ;완전한 이름사용
12                              IN  PTR    ns2.example.co.kr.    ;도메인 자동 확장 이용
```

[표 2-6] db.192.250.250에서 도메인 자동 확장 예

o @기호

도메인이름이 기원과 같다면 도메인 네임을 '@'로 나타낼 수가 있다. 이 방법은 영역 데이터 파일의 SOA 레코드에서 많이 볼 수 있으며, SOA 레코드는 다음과 같이 작성할 수 있다.

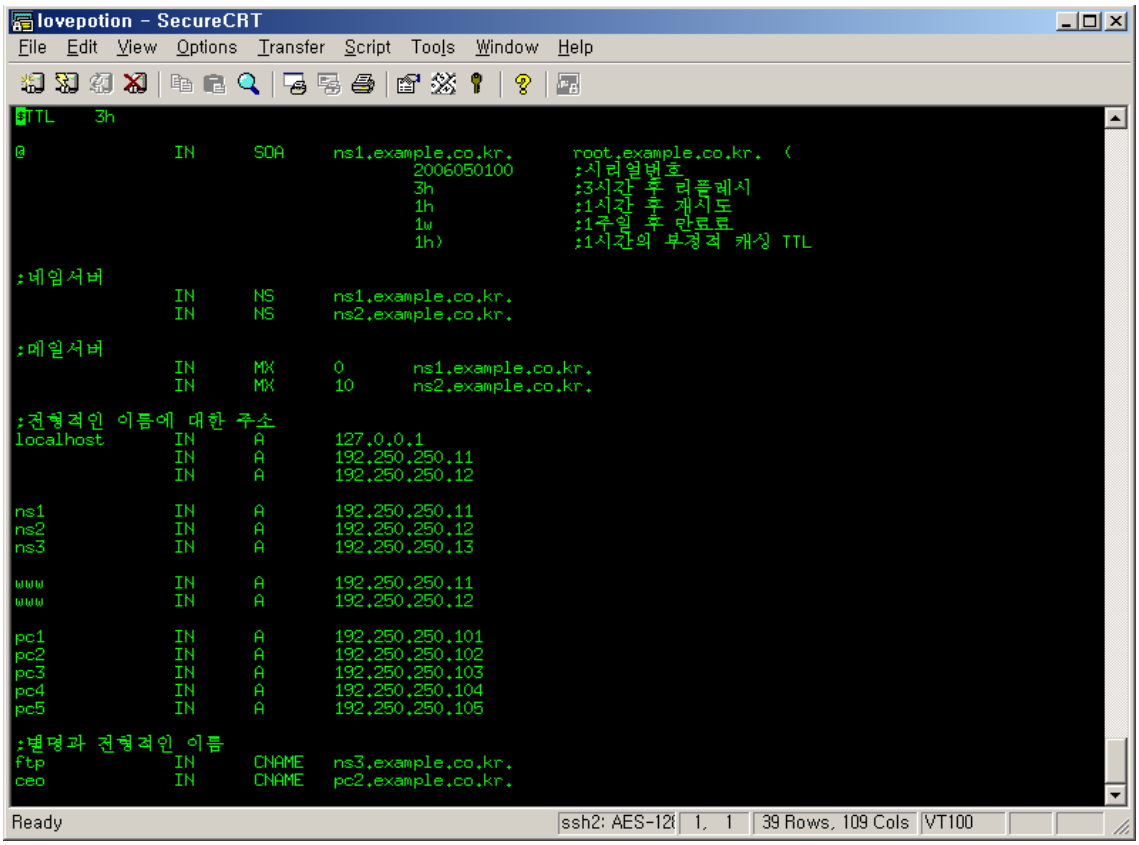
```
@  IN  SOA  ns1.example.co.kr.  root.example.co.kr.  (
                                2006050100     ;시리얼번호
                                3h             ;3시간 후 리플래시
                                1h             ;1시간 후 재시도
                                1w             ;1주일 후 만료
                                1h)            ;1시간의 부정적 캐싱 TTL
```

[표 2-7] @ 기호를 이용한 축약 예

또한 NS 레코드의 경우 @가 이미 내포되어 있으므로 도메인 이름 자체를 생략할 수 있다.

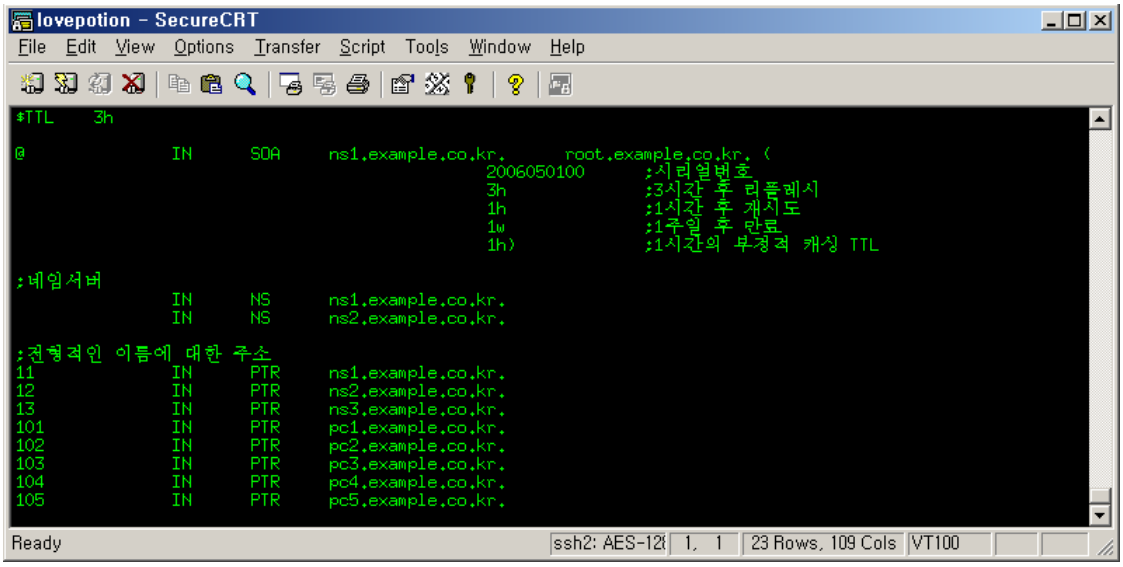
o 마지막 이름 반복

줄의 첫 열이 공백 문자이거나 탭 문자라면, 가장 최근의 리소스 레코드의 이름을 이용한다. 이것은 하나의 이름에 대하여 여러 개의 리소스 레코드들이 있는 경우에 유용하다. 이 방법은 리소스 레코드의 종류가 다르더라도 이용할 수 있다. 다음은 앞서 설명한 축약을 이용한 영역 데이터 파일 db.example.co.kr 이다.



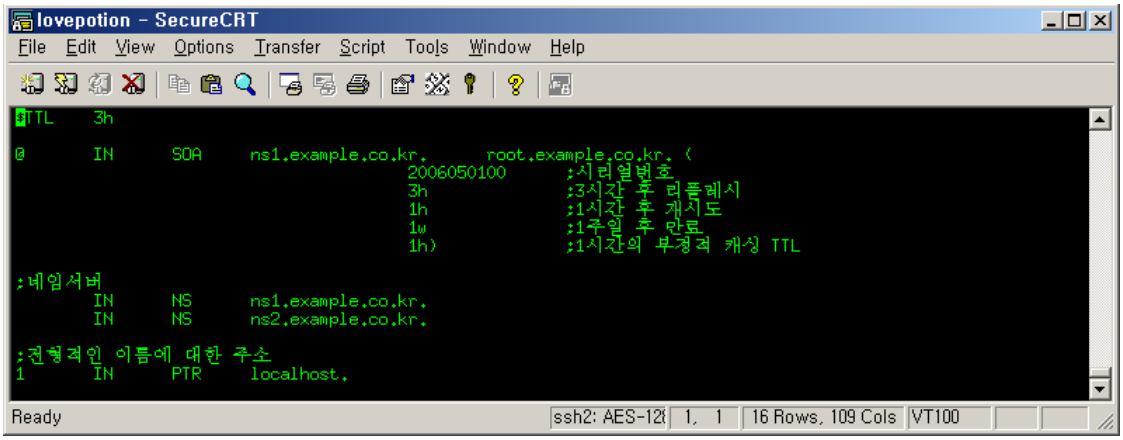
[그림 2-15] 축약을 이용한 db.example.co.kr 파일

다음은 축약을 이용한 영역 데이터 파일 db.192.250.250 이다.



[그림 2-16] 축약을 이용한 db.192.250.250 파일

다음 축약을 이용한 db.127.0.0 파일이다.



[그림 2-17] 축약을 이용한 db.127.0.0 파일

□ 호스트 이름 검사

RFC 952에서 호스트 이름과 그 외의 이름에 대한 규칙이 정의되었는데, 다음과 같다. 우선 호스트 이름은 영어 알파벳 문자와 숫자의 조합으로 이루어진다. 또한 이름 중간에 하이픈(-)이 쓰일 수 있으며, 밑줄 문자(_)는 호스트 이름으로 쓰일 수 없다. 호스트 이름이 아닌 다른 이름들은 인쇄 가능한 ASCII 문자들로 구성할 수 있다. BIND 4.9.4 이후 버전에서는 대부분 이러한 호스트 이름 검사 기능이 제공되고 있는데, 문제는 밑줄 문자를 호스트 이름의 일부로 쓰고 있는

망에서 문제가 발생 할 수 있다는 점이다.

이러한 강력한 이름 검사가 문제가 되는 경우 설정 파일을 수정하여 이러한 이름을 무시하거나 경고 메시지만 나오도록 할 수 있다. 다음은 이와 관련된 BIND 8, 9의 기본 설정 값을 보여준다.

```
options {
        check-names     master      fail;
        check-names     slave       warn;
        check-names     response    ignore;
};
```

[표 2-8] BIND 8 이후의 check-names 기본값

또한 BIND 8부터는 영역 파일별로 이를 설정할 수도 있다. 이 경우 전체적으로 설정하는 option보다는 영역에 대한 설정이 우선한다.

```
zone "example.co.kr" IN {
                        type        master;
                        file        "db.example.co.kr";
                        check-names fail;
};
```

[표 2-9] BIND 8 이후의 영역별 설정

2.4 BIND 운영

□ BIND 실행

BIND를 실행시키기 위해서는 root 권한을 가져야 한다. BIND 서버의 위치는 시스템에 따라 다를 수는 있지만 일반적으로 /usr/sbin에 있다. 그 외 가능한 디렉토리로는 /etc, /usr/etc 등이 있다.

```
#/usr/sbin/named
```

[표 2-10] BIND 실행

□ syslog 검사

BIND를 실행시킨 후 제일 먼저 확인할 것이 syslog 파일에 오류 메시지가 있는지 확인하는 것이다. 일반적으로 syslog에서 만든 파일은 /var/adm 디렉토리 또는 /var/log 디렉토리 아래 messages라는 이름으로 존재한다.
syslog는 BIND 외에도 시스템 내의 중요 사건들에 대한 기록들이 있는데, grep을 이용하여 BIND에 관련된 부분만을 확인하여 볼 수 있다.
syslog에서 BIND의 debugging 정보를 보기위해서는 /etc/syslog.conf 파일에 아래와 같이 *.info 를 추가하고 syslog 데몬을 restart 시킨다.

```
*.info;*.err;kern.debug;daemon.notice;mail.crit  /var/adm/messages
```

[표 2-11] syslog.conf 설정 예

□ nslookup 또는 dig를 이용한 검사

nslookup과 dig는 DNS를 이용하는 대표적인 클라이언트 서비스로 이를 이용하면 기본적으로 네임서버가 정상적으로 동작하는지를 확인할 수 있다. 이때 확인할 것은 지역 도메인이 정상적으로 찾아지는지, 지역도메인 내의 호스트의 주소가 정상적으로 찾아지는지, 그리고 원격 도메인 네임이 정상적으로 찾아지는지의 여부이다. 현재는 nslookup보다는 dig의 사용이 권장되고 있다.

□ 슬레이브 네임서버 운영

재해사항을 대비하기 위해 또는 네임서버를 담당하는 시스템의 부하를 줄이기 위해 슬레이브 네임서버를 운영할 수 있다. 슬레이브 서버는 보통 자신의 영역 파일을 가지지 않고 주 네임서버와의 통신을 통해 주 네임서버가 가지고 있는 영역 정보를 받아 질의의 응답을 보낸다. 주 네임서버가 아닌, 다른 슬레이브 서버가 가지고 있는 영역 정보를 얻어와 서비스를 제공하기도 한다. 과거에는 이러한 주 네임서버와 슬레이브 서버간 동기화가 주기적인 검사(polling)에 의해 이루어지는 것이 일반적이었다. 이때 참조하는 것이 SOA 레코드에 있는 값들이다.

```
example.co.kr.   IN  SOA  ns1.example.co.kr.  root.example.co.kr.  (
                                2006050100     ;시리얼번호
                                3h             ;3시간 후 리플래시
                                1h             ;1시간 후 재시도
                                1w             ;1주일 후 만료
                                1h)            ;1시간의 부정적 캐싱 TTL
```

[표 2-12] SOA RR 설정 예

처음의 시리얼 번호는 4바이트 정수 값으로 0부터 2^{32}-1(4,294,967,295)까지의 값을 가질 수 있다. 원칙적으로 1부터 시작하여 갱신이 일어날 때마다 1씩 증가시키는 것이 타당하겠지만, 일반적으로는 YYYYMMDDNN 형식으로 수정한 년도, 월, 일, 당일, 수정한 번호로 설정한다. 예를 들면 2006050102은 2006년 5월 1일 두 번째로 수정했다는 의미가 되는 것이다. 슬레이브 서버는 자신이 보관 중인 시리얼 번호와 주 네임서버의 시리얼 번호를 비교하여, 자신이 보관하고 있는 번호가 작은 경우 갱신이 일어났다고 판단하고 영역 파일을 다시 전송한다. 따라서 주 네임서버인 경우 영역 파일의 갱신이 일어날 경우 반드시 시리얼 번호를 증가시켜야 한다.

다음의 4개의 필드는 단위가 생략된 경우 초로 인식하는데, 첫 필드 값은 재생(refresh) 값으로서 슬레이브가 주 네임서버의 갱신 여부를 확인할 시간 간격을 의미한다. 일반적으로 8시간 정도가 적당하며, 거리가 멀거나 자주 변경이 일어나지 않는 경우 25시간도 고려해 볼 수 있다.

두 번째 필드는 재시도(retry) 값으로 앞서 설정된 재생 시간에 주 네임서버와 연결을 시도했는데 실패한 경우 다시 시도할 시간 간격을 뜻한다. 일반적으로 앞서 설정된 재생 값보다는 작은 것이 일반적이다.

세 번째 필드는 만료(expire) 값으로 이 시간동안 주 네임서버와 연경이 이루어지지 않으면 영역 전체의 데이터를 폐기한다. 이 필드 값은 앞서 설명한 두개의 필드 값보다는 반드시 클 필요가 있다.

네 번째 필드는 부정적 캐싱 생존시간(negative caching TTL)으로 주 네임서버로부터 부정적인 응답을 어느 정도 캐싱하고 있어야 하는지를 의미한다.

과거에는 이러한 4개의 필드에 설정하는 단위가 초만 가능했으나, 현재는 h(hour), m(minute), d(day), w(week)도 가능하다.

주 네임서버와 슬레이브 서버간 동기화 문제를 해결하기 위해 NOTIFY와 같은 방식이 새로 제공되고 있으며, 전체 영역 데이터를 전송하는 AXFR과 같은 방법 이외에도 갱신된 영역 데이터만을 전송하기 위한 새로운 방법(IXFR)도 제공되고 있다. 이에 관련된 설정은 뒤에 따로 다룬다.

□ 네임서버 제어

기존에는 스크립트 파일을 이용하여 named 프로세스를 직접 제어하거나 시그널을 이용하여 네임서버를 제어했었다. 그러나 이러한 방법으로 제어할 수 있는 기능의 제약이 문제가 되었고, 이를 극복하고자 BIND 8.2부터 특수 제어채널로 메시지를 보내 네임서버를 제어하는 방법이 등장했다. 이러한 방식으로 만들어진 것이 BIND 8.x의 ndc(name daemon controller)이고, BIND 9.x에서는 rndc (remote name daemon controller)로 개선되었다.

o BIND 8.X의 ndc

ndc를 인자없이 실행하면 ndc는 var/run/ndc라는 유닉스 도메인 소켓을 통해 로컬 호스트에서 동작 중인 네임서버와 대화하려고 하였다. 일부 운영체제는 다른 경로명에 있는 소켓을 이용하기도 하는데 이 소켓은 일반적으로 루트의 소유이며, 소유자만 읽고 쓰는 것이 가능하다. BIND 8.2 이후 버전의 네임서버는 시작 시에 이 유닉스 도메인 소켓을 생성한다. 소켓의 경로명이나 허가는 controls 구문을 이용해 바꿀 수 있다. 예를 들어, 소켓의 경로는 etc/ndc로 바꾸고, 그룹 소유권을 named로 설정하며, 소유자와 그룹 소유자 모두 소켓을 읽고 쓸 수 있게 하려면 다음을 이용하면 된다.

```
control {
        unix        /etc/ndc        perm      0660     owner   0   group 53 ; 그룹 53 named
};
```

[표 2-13] SOA RR 설정 예

이 때 사용 권한(perm)은 반드시 8진수 형태로 표시해야 한다(맨 앞의 0이 숫자가 8진수임을 나타낸다). 또한 소유자(user id)와 그룹 식별자(group id)는 모두 숫자이어야 한다. 이 유닉스 도메인 소켓은 네임서버를 제어할 권한이 있는 사용자만 접근할 수 있도록 해야 한다.

ndc는 TCP 소켓을 통해 로컬이 아닌 원격 호스트의 네임서버에서 메시지를 보낼 수도 있다. 이 기능을 이용하려면 다음의 예처럼 -c 옵션을 주고 원격 네임서버의 이름이나 주소와 제어메시지를 청취(listen)할 포트 번호를 (/)로 구별하여 옵션의 인자로 주면 된다.

```
#ndc -c 192.250.250.13/953
```

[표 2-14] TCP 소켓을 통한 원격 네임서버 제어

이렇게 ndc를 통해 원격 네임서버를 제어하려면, 제어될 네임서버에서도 설정이 필요한데, 제어될 네임서버에서는 다음과 같이 control 구문을 이용하여 특정 TCP 포트를 통해 제어 메시지를 기다리도록 설정할 수 있다.

```
controls {
        inet        *    port 953 allow { ndchost };
};
```

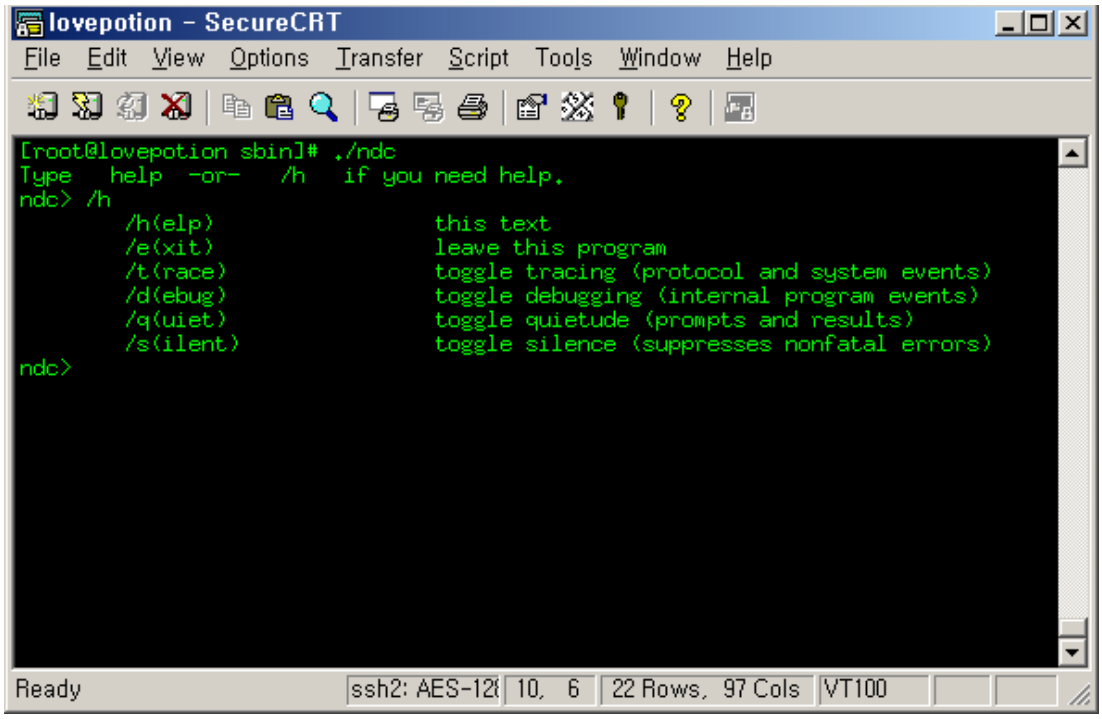
[표 2-15] 네임서버에서의 ndc를 통한 원격 제어를 위한 설정

ndc는 대화식 및 비대화식 두가지 모드로 동작한다. 비대화식 모드는 네임서버에게 시킬 일을 명령행에 지정하는 방식으로 다음은 그 예이다.

```
#ndc reload
```

[표 2-16] ndc를 통한 비대화식 제어

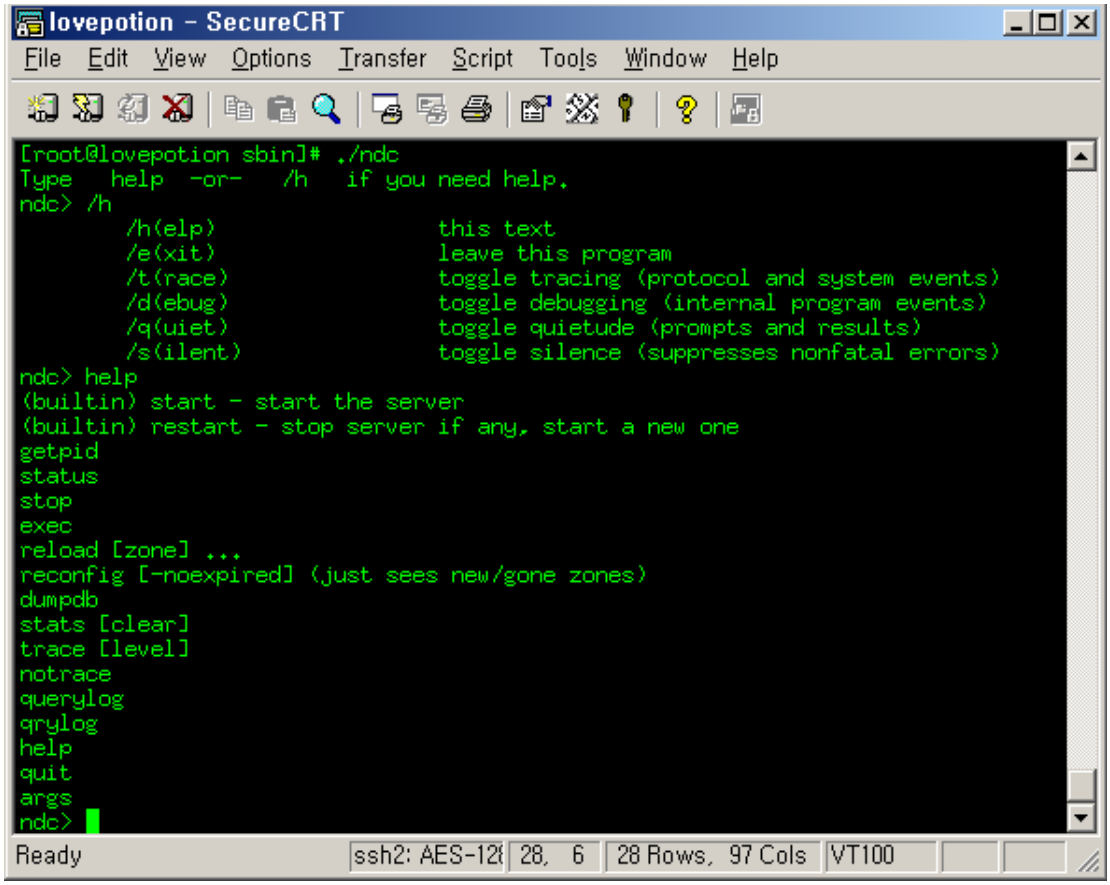
명령행에 명령을 지정하지 않으면 다음 그림과 같이 대화식 모드로 들어간다.



[그림 2-18] 대화식 ndc 실행과 /h 실행 화면

/h에서 나온 것은 ndc 자체에 대한 명령들의 도움 화면이고, ndc가 이해하는 (네임서버가 이해하는 것이 아님) 명령 목록을 나열한다. 이러한 명령들은 ndc의 동작에만 영향을 미친다. 예를 들어, /d 명령은 ndc가 디버깅 출력(예를 들어, 무엇을 네임서버에게 보내면 어떤 응답을 받는지 등)을 생성하게 한다. 그러나 네임서버의 디버깅 레벨에는 영향을 끼치지 않는다. ndc를 종료할 때에는 /e 명령을 이용한다.

실제로 네임서버를 제어할 수 있는 명령들을 알기 위해서는 help 명령을 이용한다. 여기 나열된 것들이 네임서버를 제어할 수 있는 명령들이다.



[그림 2-19] ndc에서 help 실행화면

start와 restart는 builtin 명령으로 포함되어 있다. 다음은 각 명령들의 기능이다.

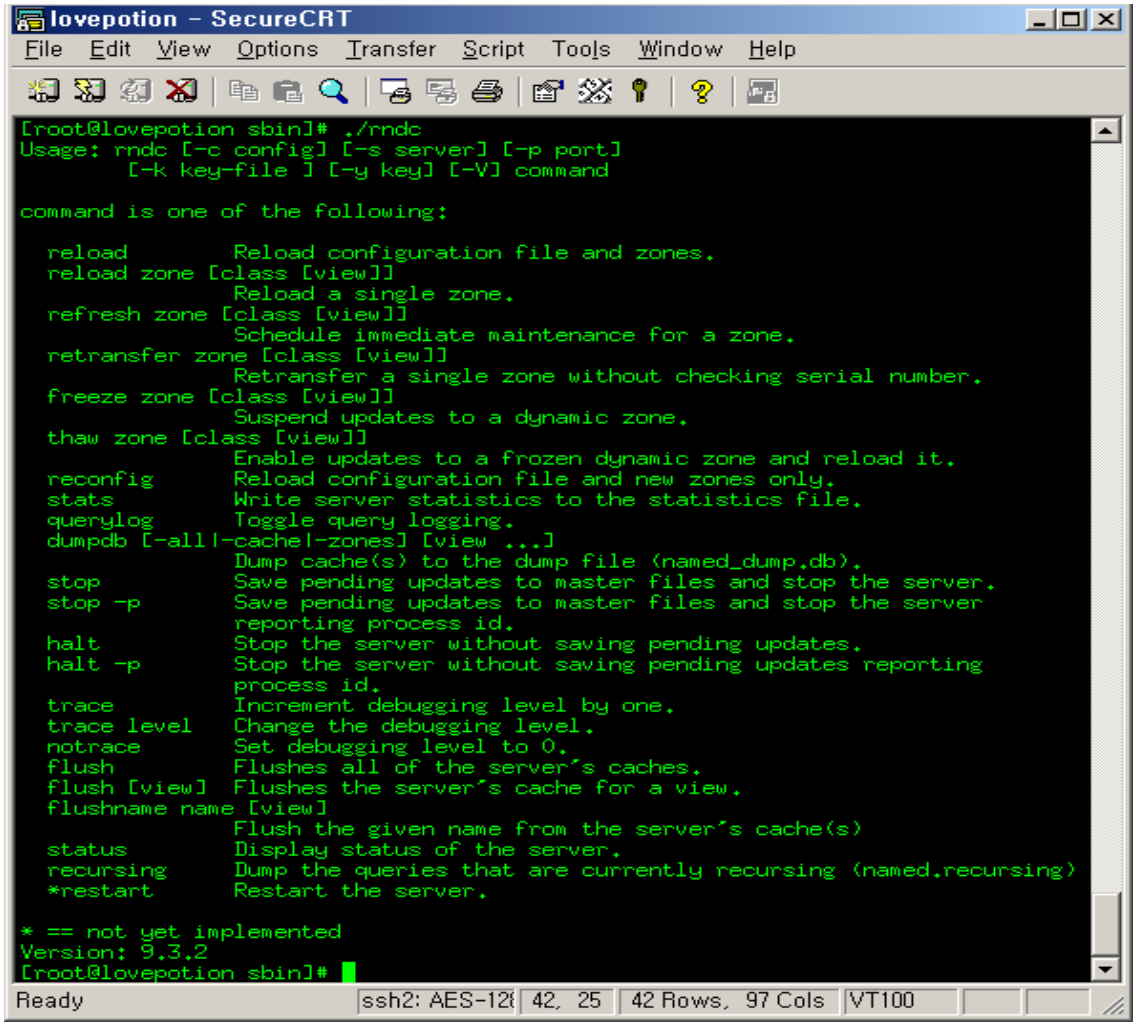
- getid : 네임서버의 현재 프로세스 ID를 출력한다.
- status : 네임서버에 관한 유용한 상태정보를 출력한다. 예를 들어, 네임서버의 버전, 디버그 레벨, 수행중인 영역 전송 개수, 질의 기록이 활성화되어 있는지 여부 등을 알려준다.
- start : 네임서버를 실행한다. named에 명령행 인자를 주고 싶다면 start 뒤에 나열하면 된다. 예를 들어, start -c /usr/local/etc/named.conf는 named -c /usr/local/etc/named.conf와 같다.
- stop : 동적 영역이 있으면 해당 영역 데이터 파일에 쓰고 네임서버를 종료한다.
- restart : 네임서버를 중지했다가 다시 시작한다. start처럼 named에 넘길 명령행 인자를 뒤에 나열할 수 있다.
- exec : 네임서버를 중지했다가 다시 시작시킨다. restart와 달리 named에 넘길 명령행 인자를 지정할 수 없다. 새로 시작하는 네임서버는 현재 돌고 있는 네임서버의 명령행 인자를 그대로 받는다.

- reload : 네임서버를 리로드(reload)한다. 주 마스터 네임서버에서 환경설정 파일을 수정하거나 영역 데이터 파일을 하나이상 고친 수에 이 명령어를 실행하도록 하자. BIND 4.9 이후 버전의 슬레이브 네임서버에 이 명령어를 실행시키면 최신이 아닌 슬레이브 영역을 업데이트한다. 그리고 reload 뒤에 도메인 네임(들)을 지정하면 네임서버는 그 영역(들)만 다시 읽어드린다.
- reconfig : 네임서버가 환경 설정 파일을 검사해 새로 추가된 영역이나 삭제된 영역이 있는지 조사하도록 지시한다. 기존의 영역 데이터는 고치지 않고 영역을 새로 추가하거나 삭제만 한 경우라면 이 명령어를 네임서버로 전달하도록 하자. 뒤에 -noexpired 플래그를 지정하면 기한 만료된 영역에 대해서도 네임서버가 에러메시지를 출력하지 않는다. 수천 개의 영역을 갖는 네임서버에 이 플래그를 적용하면 이미 알고 있는 만료 에러메시지를 보지 않아도 되기 때문에 유용하다.
- dumpdb : 네임서버의 내부 데이터베이스를 named_dump.db 라는 파일을 복사한다. 이 파일의 위치는 네임서버의 현재 디렉토리이다.
- stats : 네임서버의 통계 자료를 named.stats라는 파일로 붙여넣는다. 이 파일의 위치는 네임서버의 현재 디렉토리이다.
- trace[level] : 디버깅 정보를 named.run이라는 파일로 붙여넣는다. 이 파일의 위치는 네임서버의 현재 디렉토리이다. 레벨의 값을 높게 줄수록 더 자세한 디버깅 정보를 기록한다.
- notrace : 디버깅을 하지 않도록 한다.
- querylog(또는 qrylog) : 모든 질의를 syslog에 기록할 것인지 아니면 토글(toggle)한다. 즉, 이 명령어를 한번 보내면 모든 질의를 syslog에 기록하고, 한번더 보내면 기록을 중지한다. 그리고 이러한 내용은 LOG_INFO 우선순위로 기록된다. named는 QRYLOG를 정의한 상태로 (기본적으로 정의되어 있음) 컴파일
- quit : 제어 세션을 종료한다.

o BIND 8.X의 rndc

rndc도 BIND 8.X의 ndc와 같이 네임서버를 제어하는 기능을 하는데, ndc와 비교하여 가장 중요한 변화는 ndc의 경우 유닉스 도메인 소켓을 이용했는데, rndc는 유닉스 도메인 소켓이 아닌, inet 도메인 소켓을 이용한다는 것이다.

이러한 inet 도메인 소켓의 사용은 원격 네임서버의 제어도 가능하게 한다. 그리고 보안 강화를 목적으로 키 값을 요구한다. 따라서 서로간의 통신에 이용할 비밀키를 설정하는 것이 필요하며 이러한 설정을 위한 rndc.conf 파일을 요구한다. 또한 ndc에서는 명령행에서 명령없이 실행하면 대화식으로 실행되었는데, rndc에서는 명령들을 소개하는 간단한 화면만 나타나고 바로 종료된다. 즉 대화식 실행이 없다. 다음은 명령행에 명령없이 rndc를 실행한 화면이다.



[그림 2-20] rndc에서 명령 없이 실행한 화면

BIND 9.X 초기에는 ndc에서 제공하고 있는 기능의 일부분만 지원되었으나 지금은 대부분의 rndc에서도 제공되고 있다. 또한 아래와 같은 새로운 기능이 제공되고 있다.

- refresh : 슬레이브 영역을 즉시 점검하도록 한다.
- halt : 대기중인 업데이트 내용을 파일에 기록하지 않고 네임서버를 종료한다.

BIND 9.X 네임서버에서는 rndc가 이용할 제어 채널의 디폴트 포트 번호가 953으로 설정되어 있으므로, 포트 번호를 생략할 수 있다. 그럴 경우 포트 953을 기본으로 청취하며, 다음과 같이 keys를 설정해야 한다.

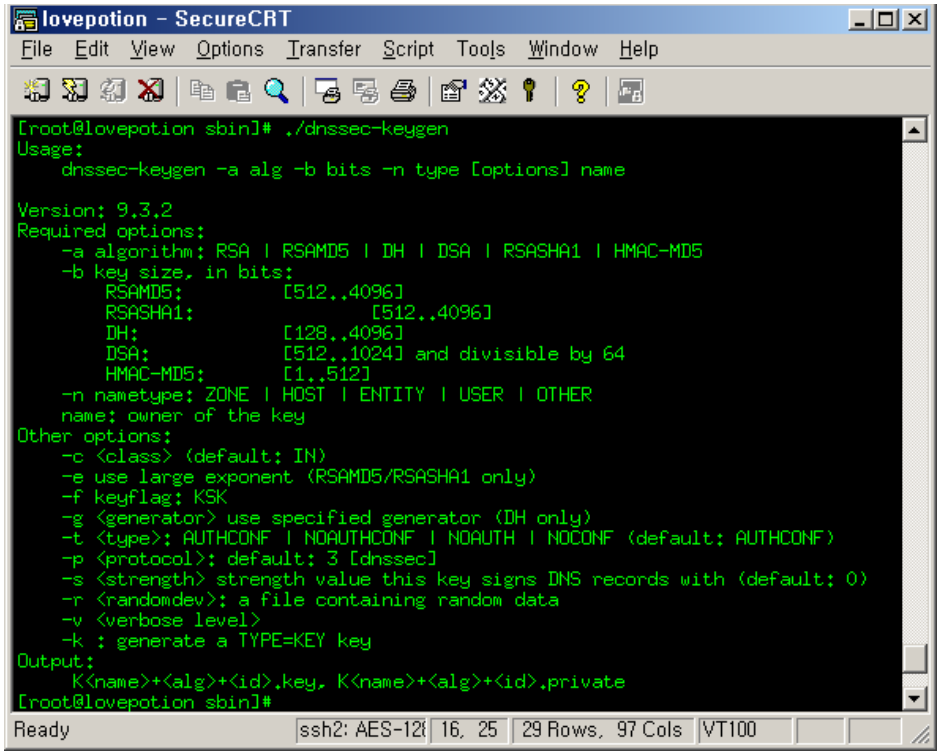
```
controls {
        inet      *   allow { any; } keys { "rndc-key"; };
};
```

[표 2-17] 네임서버에서의 rndc를 통한 원격 제어를 위한 설정

```
key "rndc-key" {
          algorithm     hmac-md5;
          secret        "Zm9vCg==";
};
```

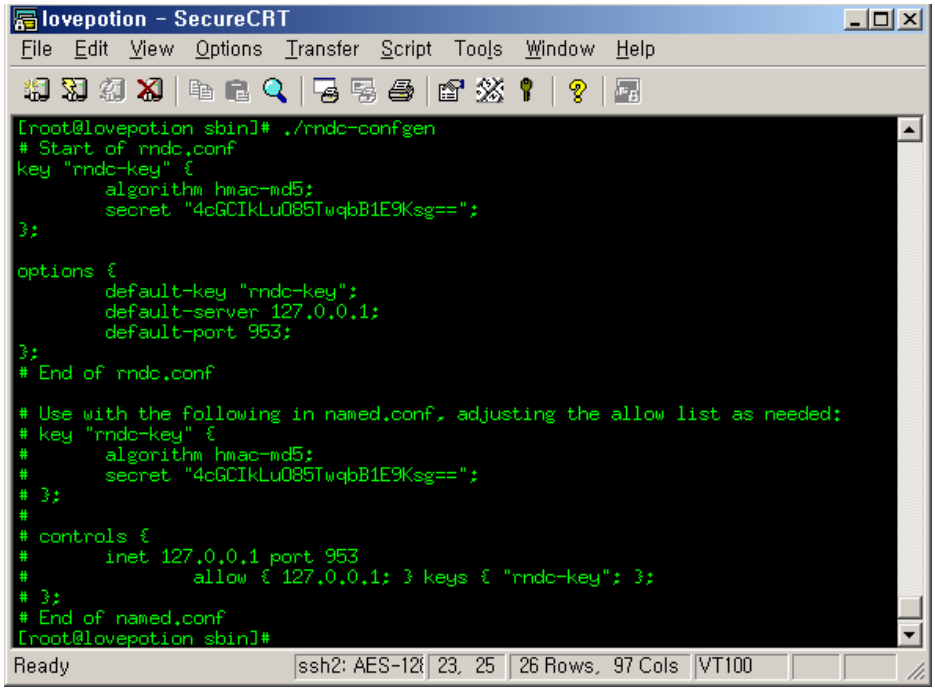
[표 2-18] key 구문 예

위의 key 구문을 named.conf 파일에 직접 포함시킬 수도 있지만 named.conf 파일을 아무나 읽을 수 있도록 허가한다면 보안상의 문제가 될 수도 있기 때문에, 읽기 권한을 제한한 별도의 파일로 만든 후에 INCLUDE 문을 이용하여 named.conf에 포함시키는 방법이 좋다. 이러한 키 알고리즘은 현재 HMAC-MD5 알고리즘만 지원된다. 이 알고리즘은 빠른 MD-5 보안 해시(hash) 알고리즘을 이용해 인증을 수행한다. 비밀값은 암호를 BASE 64 로 인코딩한 것이며, named 및 인가된 rndc 사용자들이 그 암호를 공유한다. 비밀값을 생성하려면 mmencode나 BIND 배포본에 포함되어 있는 dnssec-keygen을 이용하면 된다. 다음은 인자없이 dnssec-keygen을 실행하여 도움말로 출력하도록 한 것이다.



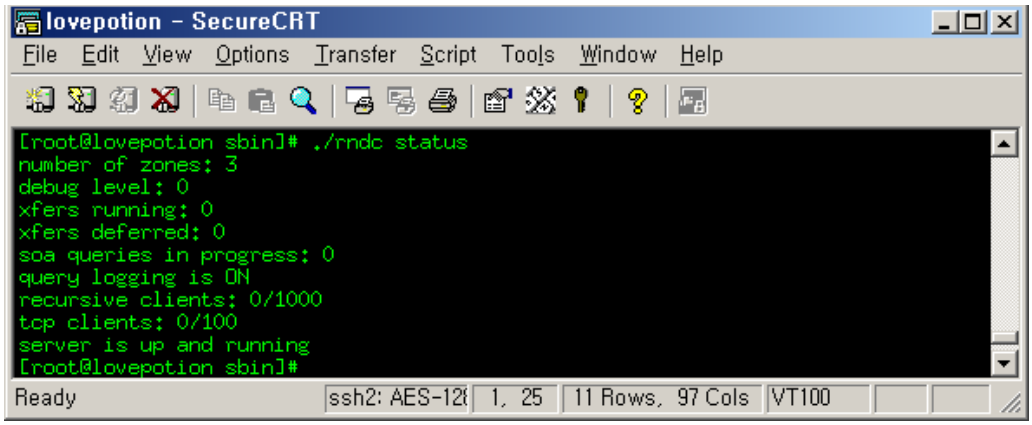
[그림 2-21] dnssec-keyggen 도움말 출력 화면

앞서 설명한 named.conf와 rndc.conf 파일 생성을 돕는 명령어가 제공되는데, 그것은 rndc-confgen을 실행시킨 화면이다.



[그림 2-22] rndc-confgen 실행 화면

BIND의 버전이 9.2 이후라면 rndc-confgen -a를 이용하여 rndc와 named가 공통으로 이용할 키 파일을 생성하도록 할 수 있다. 이때 named.conf 내에 controls 문장이 없어야 하고, rndc.conf 파일은 존재하면 안된다. 둘 중 하나라도 있으면 자동 설정이 되지 않는다. 다음은 rndc로 현재 네임서버 상태를 확인한 화면이다.



[그림 2-23] rndc status 실행화면

네임서버를 여러 개 제어할 것이라면 각각의 네임서버에 서로 다른 키를 연계시킬 수 있다. 이때에는 각 키를 별도의 key 구문으로 정의하고 서로 다른 server 구문으로 연계하면 된다. 그런 후에 rndc에 -s 옵션으로 제어할 서버를 지정해 실행한다. 또한 아직 특정한 네임서버에 연계시키지 않은 키가 있다면 명령행에 -y 옵션을 이용해 특정키를 지정할 수 있다. 마지막으로, 네임서버가 제어 메시지를 위해 (953번 이외의) 비표준 포트를 청취하는 상황이라면 -p 옵션을 이용해 어떤 포트로 연결해야 할지 rndc에게 알려주어야 한다.

3장 Windows 2003 DNS 서버

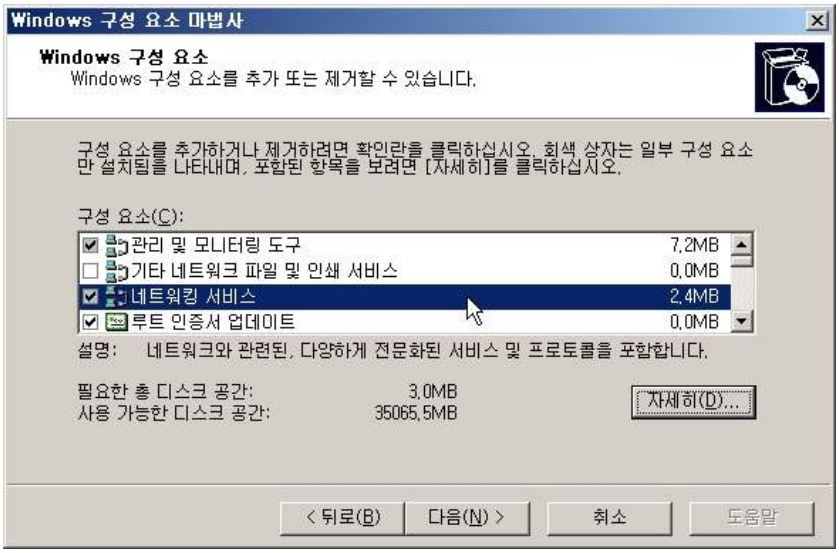
3.1 Windows 2003 DNS 서버 설치

Windows 2003 DNS server는 GUI 방식으로 다음과 같은 과정에 따른다.

□ Windows 2003 DNS 서버 설치 과정

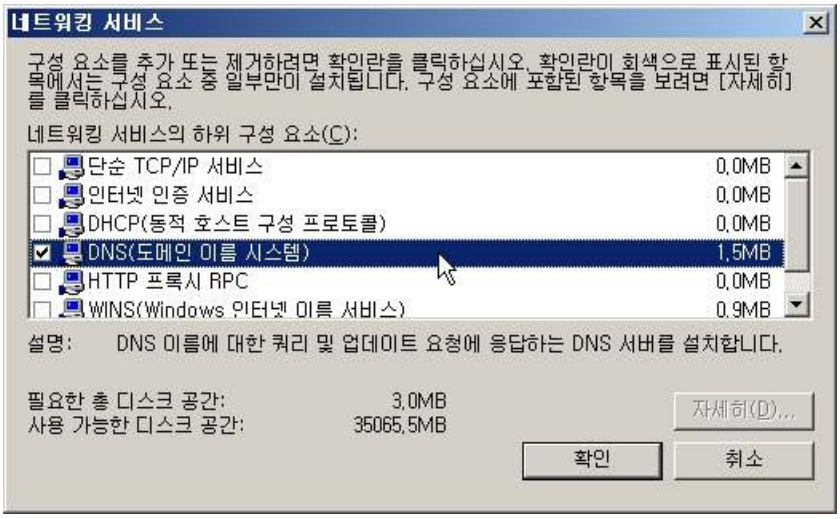
o Windows 구성 요소 마법사를 연다.
- 시작, 제어판을 차례로 누른 다음 프로그램 추가/제거를 클릭한다.
- Windows 구성 요소 추가/제거를 클릭한다.

o 구성 요소에서 **네트워킹 서비스** 확인란을 선택한 다음 **자세히**를 클릭한다.



[그림 3-1] Windows 구성 요소 마법사

o 네트워킹 서비스의 하위 구성 요소에서 **DNS(도메인 이름 시스템)** 확인란을 선택하고 확인을 누른 후 다음을 클릭한다.



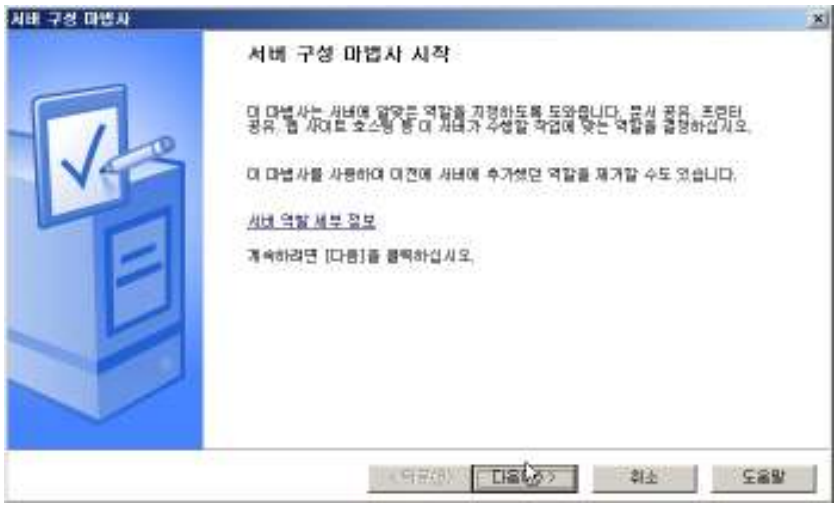
[그림 3-2] 네트워킹 서비스

o 프롬프트가 나타나면 복사할 파일 위치에 배포 파일의 전체 경로를 입력한 다음 **확인**을 누른다.

3.2. 윈도우 2003 DNS 서버 구성

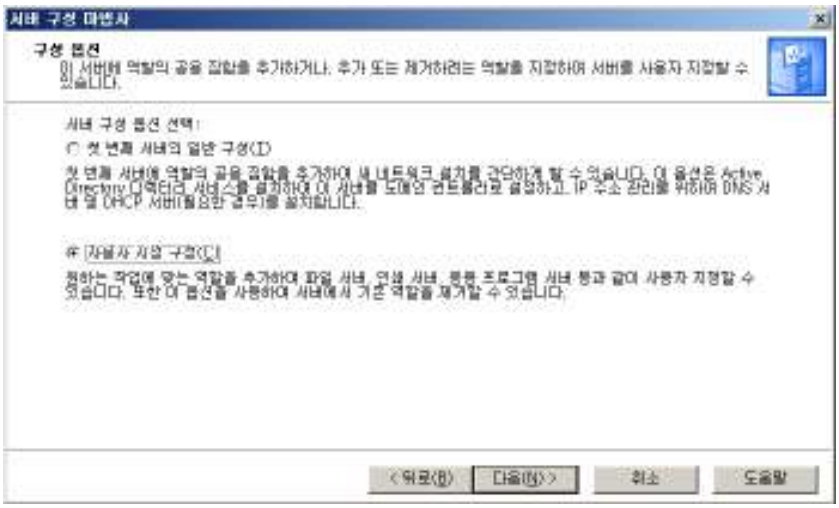
□ 서버구성 마법사 사용

o 시작을 누르고 프로그램, 관리 도구를 차례로 가리킨 다음 **서버 구성 마법사**를 클릭한다.



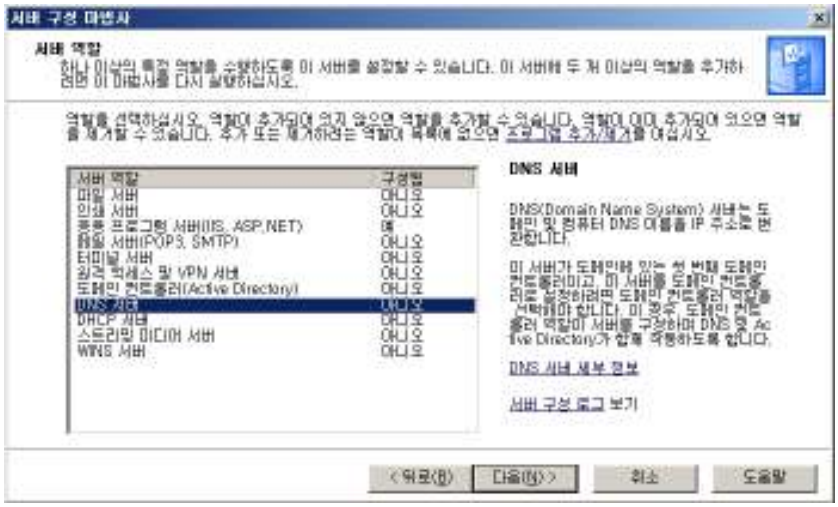
[그림 3-3] 서버 구성 마법사 시작

- 구성 옵션 선택 화면에서 **사용자 지정 구성**을 선택한다.



[그림 3-4] 구성 옵션

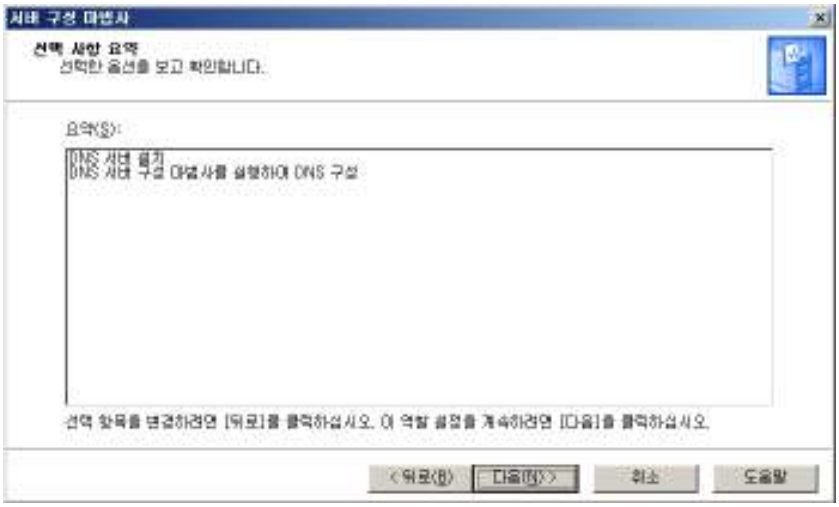
- 서버 역할 페이지에서 **DNS 서버**를 누르고 다음을 누른다.



[그림 3-5] 서버 역할

o 선택 사항 요약 페이지에서 선택한 옵션을 보고 확인한다. 다음 항목이 이 페이지에 있어야 한다.

- DNS 서버 설치
- DNS 서버 구성 마법사를 실행하여 DNS 구성



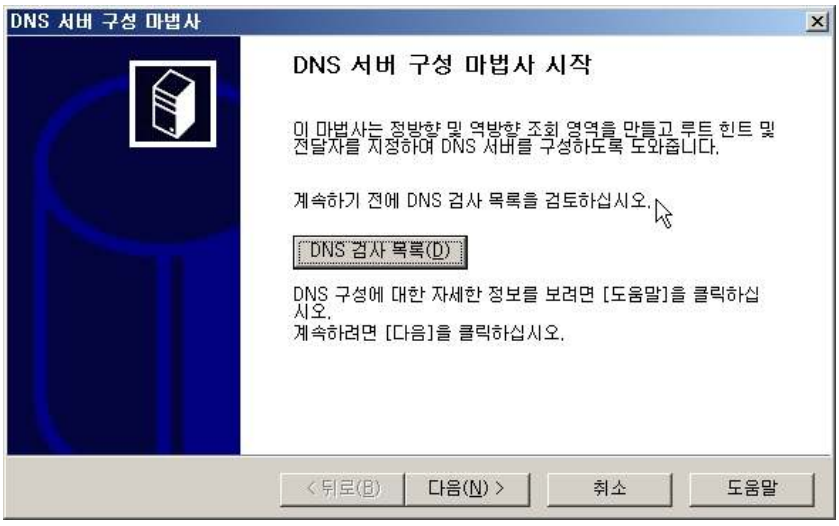
[그림 3-6] 선택 사항 요약

o 선택 사항 요약 페이지에 이러한 두 항목이 나타나면 다음을 누른다. 선택 사항 요약 페이지에 이러한 두 항목이 나타나지 않으면 뒤로를 눌러 서버 역할 페이지로 돌아가 DNS를 누르고 다음을 클릭한다.

o 서버 구성 마법사가 DNS 서비스를 설치하면 먼저 이 서버의 IP 주소가 정적 주소인지, 자동으로 구성되었는지 확인한다. 서버가 현재 IP 주소를 자동으로 얻도록 구성된 경우 Windows 구성 요소 마법사의 구성 요소 구성 중 페이지에 이 서버를 정적 IP 주소로 구성할 것인지 묻는 메시지가 나타난다. 이렇게 하려면 다음과 같이 한다.

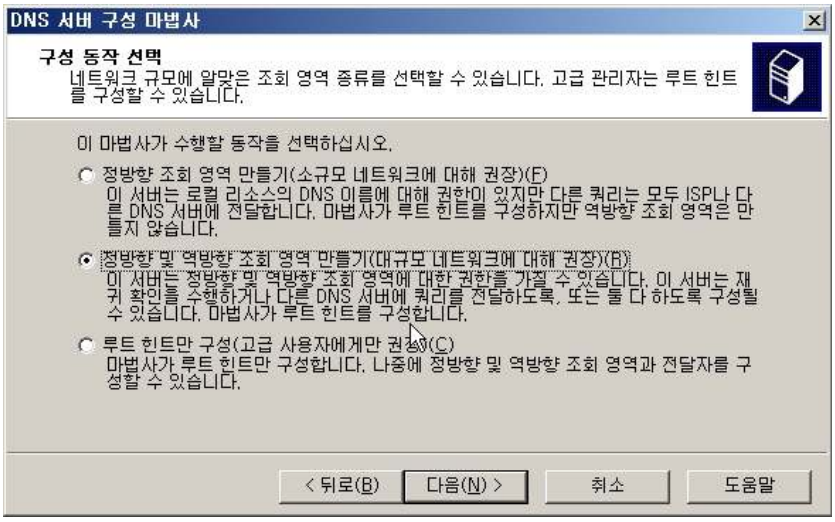
- 로컬 영역 연결 속성 대화 상자에서 인터넷 프로토콜(TCP/IP)을 누른 다음 속성을 누른다.
- 인터넷 프로토콜(TCP/IP) 등록 정보 대화 상자에서 다음 IP 주소 사용을 누른 다음 이 서버의 정적 IP 주소, 서브넷 마스크 및 기본 게이트웨이를 입력한다.
- 기본 설정 DNS 서버에 이 서버의 IP 주소를 입력한다.
- 보조 DNS 서버에 다른 내부 DNS 서버의 IP 주소를 입력하거나 비워둔다.
- 사용자는 DNS의 정적 주소 설정을 마치면 확인을 누른 다음 닫기를 누른다.

o 닫기를 누르면 **DNS 서버 구성 마법사**가 시작 마법사에서 다음 단계를 수행한다.



[그림 3-7] DNS 서버 구성 마법사

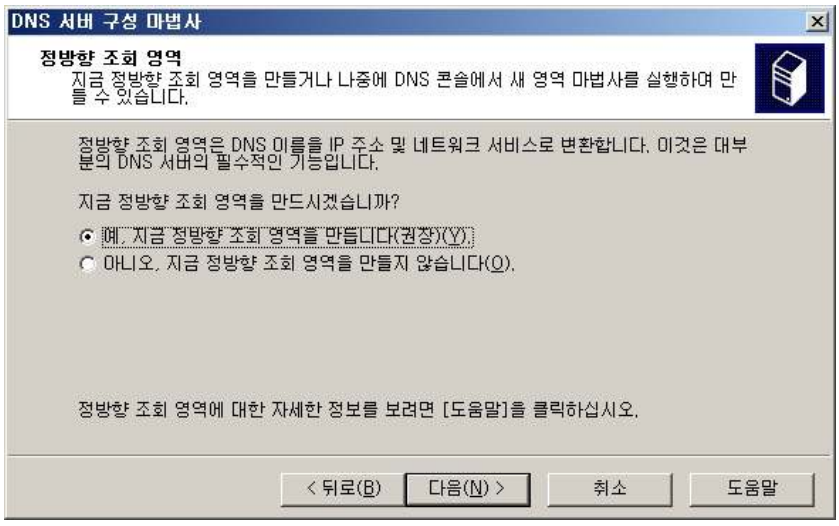
- 구성 동작 선택 페이지에서 **정방향 및 역방향 조회 영역 만들기** 확인란을 선택하고 **다음**을 누릅니다.



[그림 3-8] 구성 동작 선택

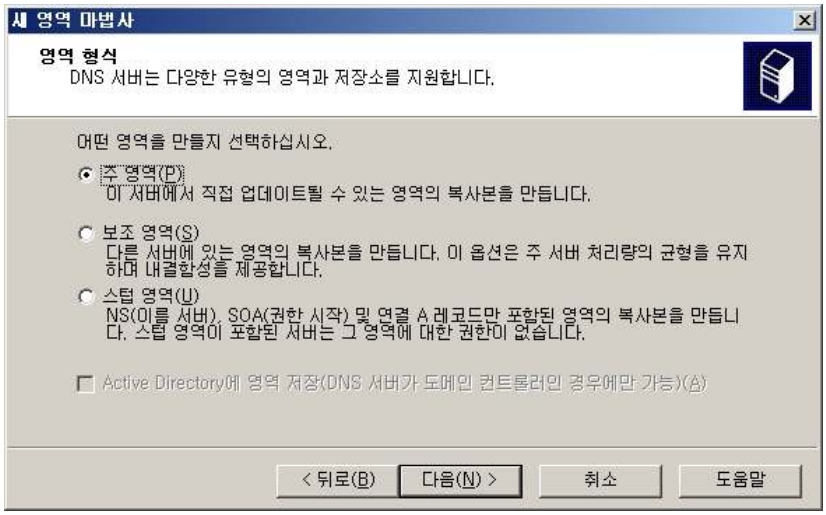
o 정방향 조회 영역 생성

- 정방향 조회 영역 생성을 위해 **예**를 선택하고 **다음**을 클릭한다.



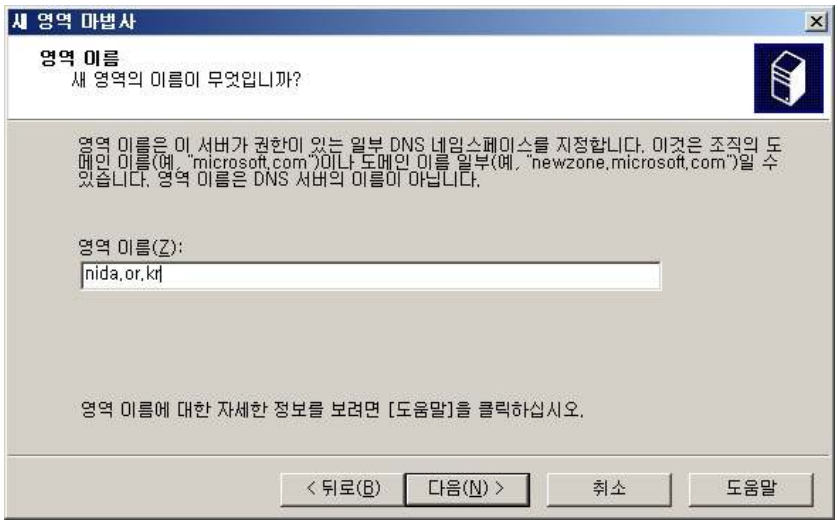
[그림 3-9] 정방향 조회 영역

- 이 DNS가 네트워크 리소스에 대한 DNS 리소스 레코드가 들어 있는 DNS 영역을 호스팅하도록 지정하려면 **주 영역**을 선택하고 **다음**을 클릭한다.



[그림 3-10] 영역 형식

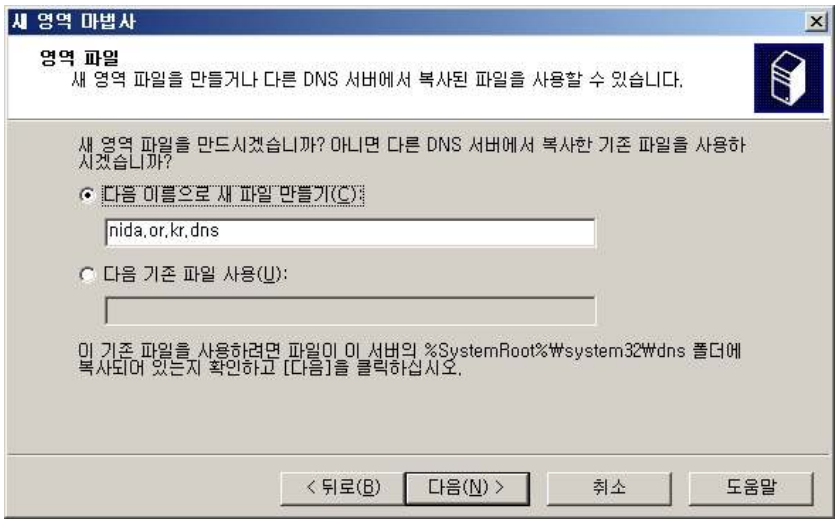
- 영역 이름 페이지의 영역 이름에서 네트워크에 대한 DNS 영역 이름을 지정하고 다음을 클릭한다.
- 영역 이름은 소규모 조직이나 지사용 DNS **도메인의 이름**과 같다. (예: nida.or.kr)



[그림 3-11] 영역 이름

- 영역 파일을 생성한다.

· 기본값으로 도메인이름.dns 이름으로 파일이 생성한다. (DNS 데이터베이스는 %systemroot%₩system32₩dns 폴더에서 확인이 가능하다.)



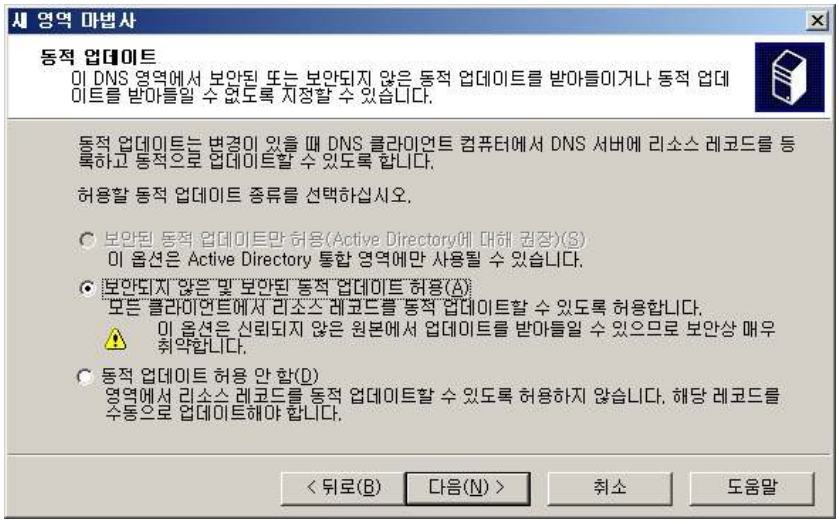
[그림 3-12] 영역 파일

※ nida.or.kr.dns 영역 파일

```
;  Database file nida.or.kr.dns for nida.or.kr zone.
;      Zone version:  1
;

@                       IN  SOA nida-dns.  hostmaster. (
          1             ; serial number
          900           ; refresh
          600           ; retry
          86400         ; expire
          3600        ) ; default TTL

;
;  Zone NS records
;

@                       NS          nida-dns.

;
;  Zone records
;
```

- 동적 업데이트 페이지에서 **보안되지 않은 및 보안된 동적 업데이트 허용**을 누르고 **다음**을 누른다.

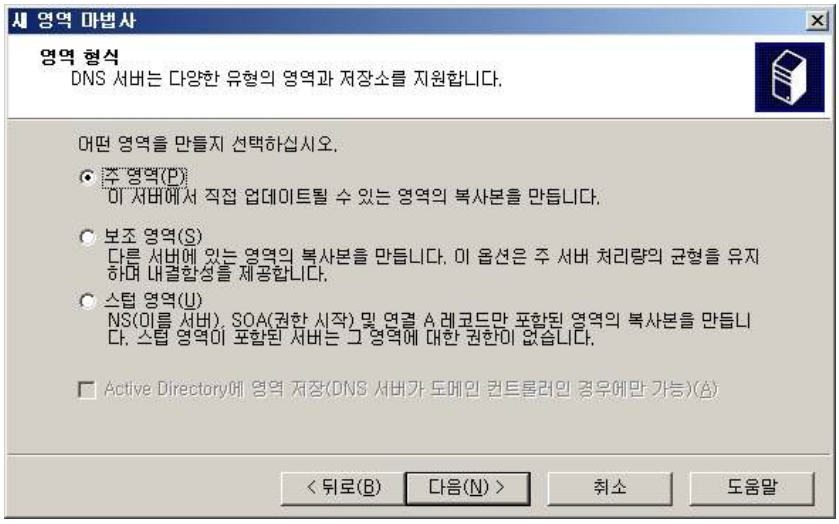
· 이렇게 하면 네트워크에 있는 리소스에 대한 DNS 리소스 레코드가 자동으로 업데이트 된다.



[그림 3-13] 동적 업데이트

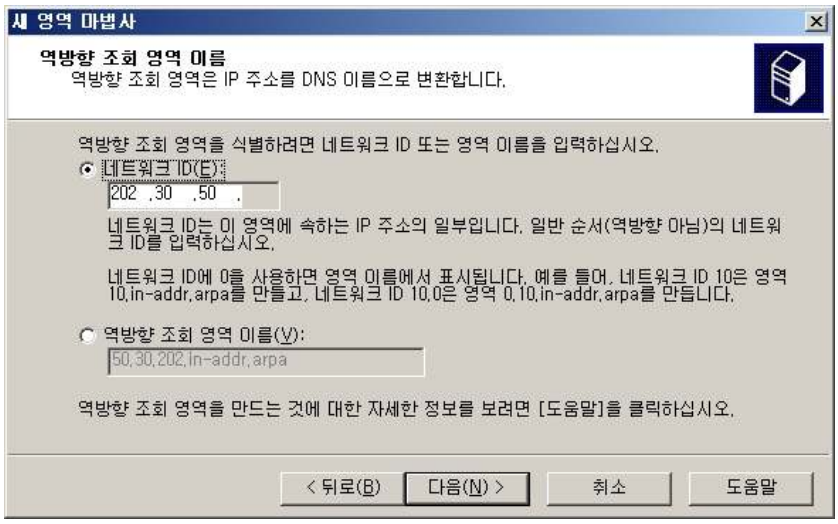
o 역방향 조회 영역 생성

- 역방향 조회 영역 생성을 위해 **예**를 선택하고 다음을 누른다.



[그림 3-14] 영역 형식

- 역방향 조회 영역 이름을 설정한다.



[그림 3-15] 역방향 조회 영역 이름

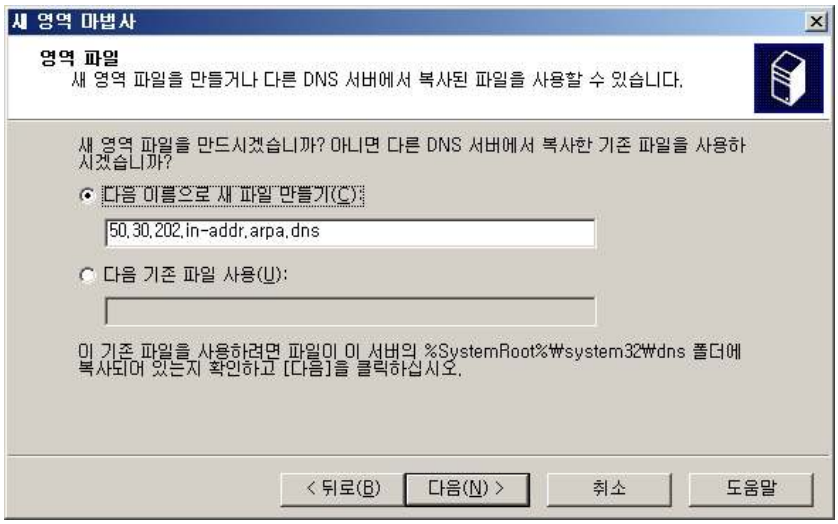
네트워크 ID에는 인터넷에서 사용할 수 있도록 공인 받은 IP 주소에 대한 정보를 사용하여, 클래스에 따라 입력해야 합니다.

역방향 조회 영역의 네트워크 ID 입력은 아래와 같은 규칙을 사용합니다. 입력값은 IP 주소가 w.x.y.z 일때 IP 주소중 어떤 부분을 입력시켜야 하는지를 나타냅니다.

클래스 구분	클래스의 IP 주소 범위	입력값
A 클래스	1.0.0.1~126.255.255.253	IP 주소의 앞 8자리 (w)
B 클래스	128.0.0.1~191.255.255.254	IP 주소의 앞 16자리 (w.x)
C 클래스	192.0.0.1~223.255.255.254	IP 주소의 앞 24자리 (w.x.y)

[표 3-1] 역방향 조회 영역의 네트워크 ID 입력 규칙

- 영역 파일을 생성한다.



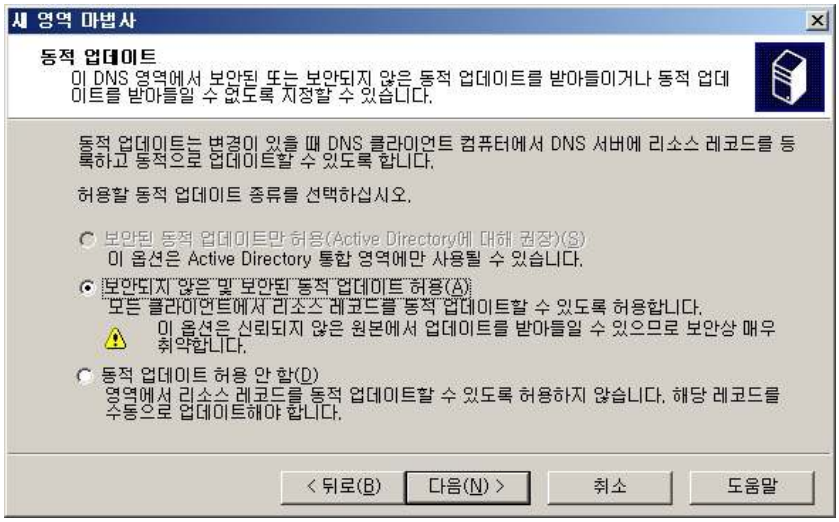
[그림 3-16] 영역 파일

※ 50.30.202.in-addr.arpa.dns 영역 파일

```
;  Database file 50.30.202.in-addr.arpa.dns for 50.30.202.in-addr.arpa zone.
;      Zone version:  1
;

@                       IN  SOA nida-dns.  hostmaster. (
          1             ; serial number
          900           ; refresh
          600           ; retry
          86400         ; expire
          3600        ) ; default TTL

;
;  Zone NS records
;

@                       NS          nida-dns.

;
;  Zone records
;
```

- 동적 업데이트 페이지에서 보안되지 않은 및 보안된 동적 업데이트 허용을 누르고 다음을 누른다.

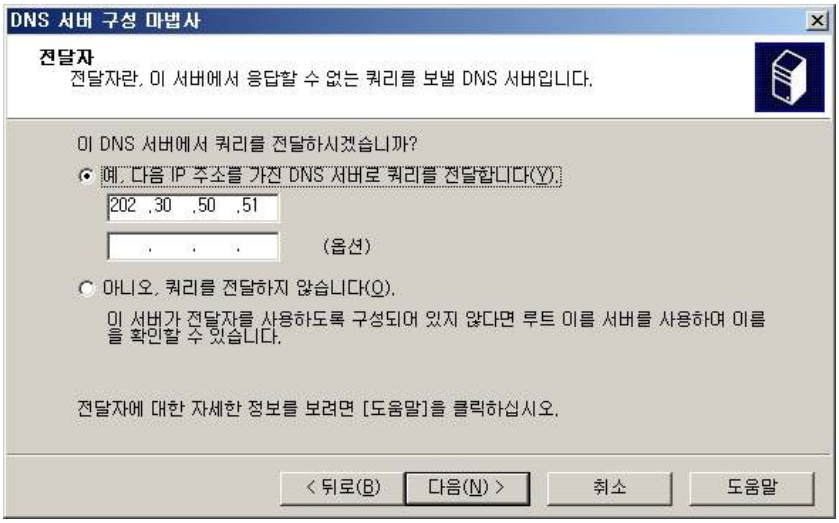
· 이렇게 하면 네트워크에 있는 리소스에 대한 DNS 리소스 레코드가 자동으로 업데이트 된다.



[그림 3-17] 동적 업데이트

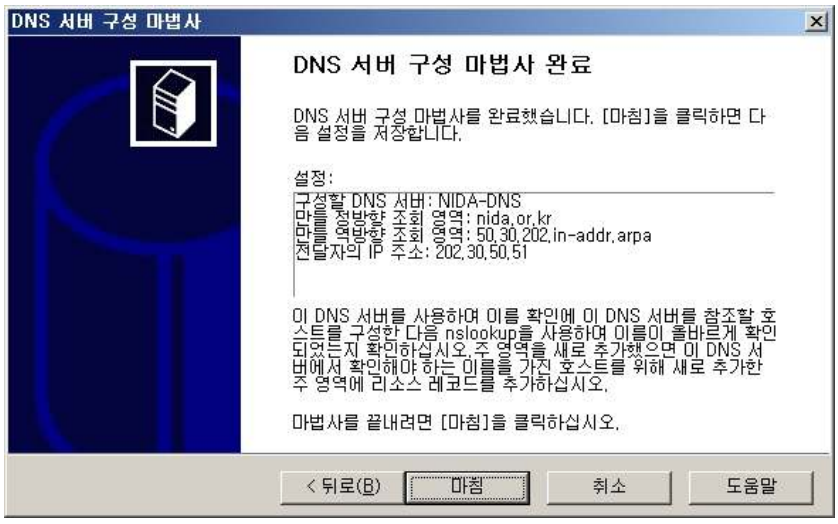
- 전달자 페이지에서 **예, 다음 IP 주소를 가진 DNS 서버로 쿼리를 전달합니다.**를 누르고 다음을 클릭한다.

· 이 구성을 선택하면 네트워크 외부의 DNS 이름에 대한 모든 DNS 쿼리가 사용 중인 ISP 또는 본사에 있는 DNS로 전달된다. ISP 또는 본사 DNS 서버가 사용하는 IP 주소를 하나 이상 입력한다.



[그림 3-18] 전달자

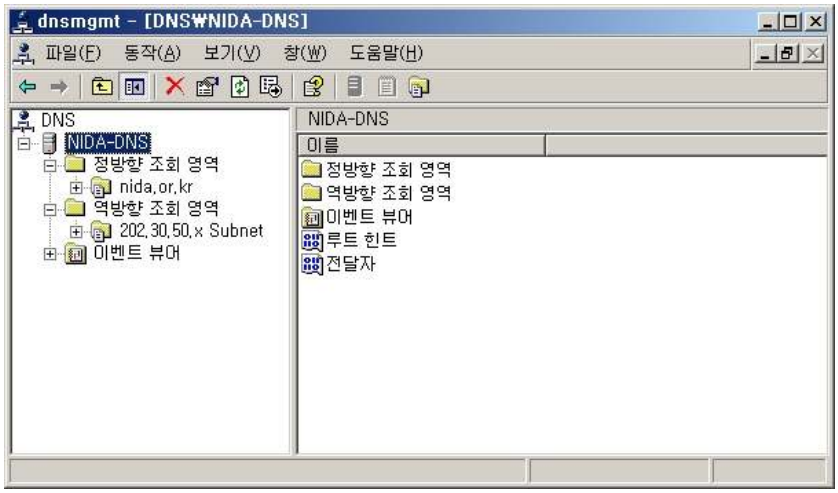
- DNS 서버 구성 마법사의 DNS 서버 구성 마법사 완료 페이지에서 뒤로를 눌러 설정을 변경할 수 있습니다. 선택 사항을 적용하려면 마침을 누릅니다.



[그림 3-19] DNS 서버 구성 마법사 완료

DNS 서버 구성 마법사를 마치고 나면 서버 구성 마법사는 이 서버는 이제 DNS 서버로 작동한다.

서버 구성 마법사에서 서버에 대한 모든 변경 사항을 검토하거나 새 역할이 성공적으로 설치되었는지 확인하려면 서버 구성 로그를 누른다. 서버 구성 마법사 로그는 %systemroot%₩Debug₩Configure Your Server.log에 있습니다. 서버 구성 마법사를 닫으려면 **마침**을 누른다.



[그림 3-20] 윈도우 2003 DNS 서버 구성 완료

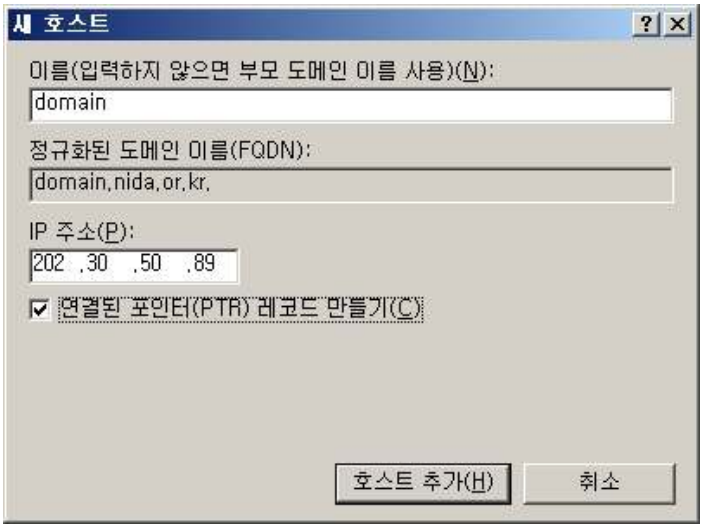
3.3. 윈도우 2003 DNS 서버 설정

윈도우 2003 DNS 서버 구성이 끝나면, 웹서버, 메일서버, FTP 서버 등 실제 외부에서 접근해야 할 서버들을 도메인에 추가한다. 도메인에 등록되는 위와 같은 서버들의 정보를 레코드(record)라고 부른다.

□ 영역에 호스트(A) 리소스 레코드 추가

o DNS 콘솔 트리에서 해당 정방향 조회 영역을 마우스 오른쪽 단추로 클릭한 다음 **새 호스트(A)**를 클릭한다.

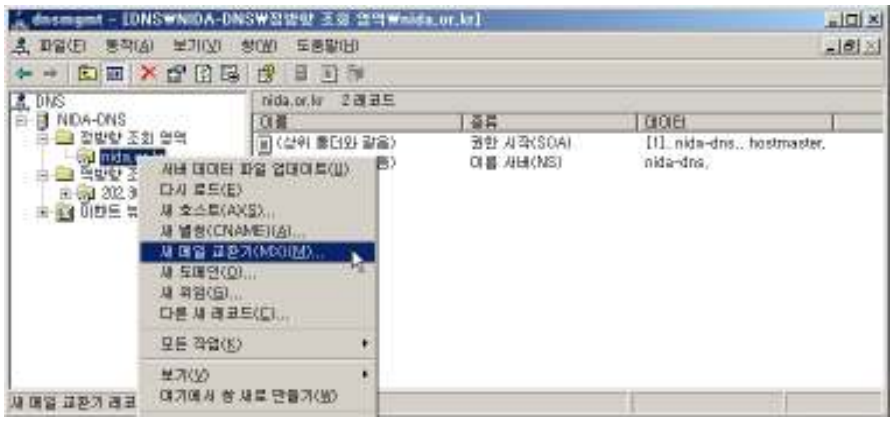
o 이름 입력란에 새 호스트의 DNS 컴퓨터 이름을 입력한다.
- IP 주소 입력란에 새 호스트의 IP 주소를 입력합니다.
- 선택 사항으로 **연결된 포인터(PTR) 레코드 만들기** 확인란을 선택하여 이름 및 IP 주소에 입력한 정보를 기반으로 이 호스트의 역방향 영역에 추가 포인터 레코드를 만듭니다.
  ※ 영역에 A 리소스 레코드를 추가할 때 자동으로 만들어진 PTR 리소스 레코드는 해당 A 리소스 레코드가 삭제되면 자동으로 삭제됩니다.
- 호스트 추가를 클릭하여 영역에 새 호스트 레코드를 추가합니다.



[그림 3-21] 새 호스트

□ 영역에 MX(메일 교환기) 리소스 레코드 추가

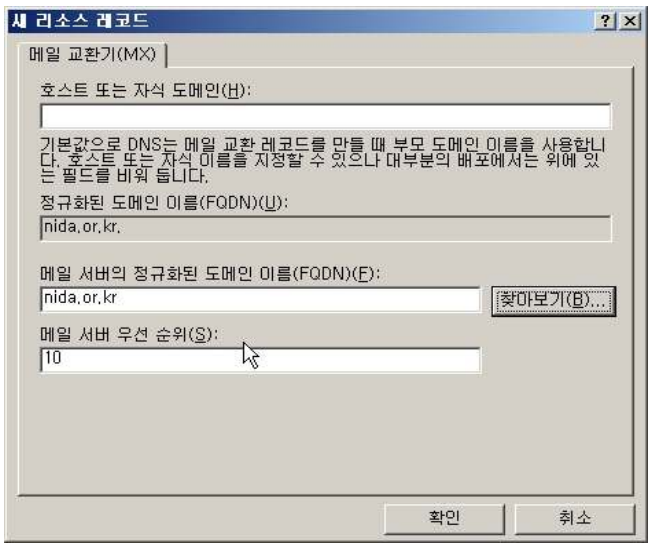
o DNS 콘솔 트리에서 해당 정방향 조회 영역을 마우스 오른쪽 단추로 클릭한 다음 새 메일 교환기(MX)를 클릭한다.



[그림 3-22] 새 메일 교환기(MX)

o 이 레코드를 사용하여 메일을 배달할 도메인 이름을 호스트 또는 자식 도메인 입력란에 입력한다.

- 지정된 도메인 이름에 메일을 배달하는 메일 교환기 또는 메일 서버 호스트의 DNS 호스트 컴퓨터 이름을 메일 서버의 정규화된 도메인 이름(FQDN) 입력란에 입력한다.
- 옵션으로 찾아보기를 클릭하여 이 도메인에서 이미 호스트(A) 레코드가 정의된 메일 교환기 호스트의 DNS 네임스페이스를 볼 수 있다.



[그림 3-23] 메일 교환기 설정

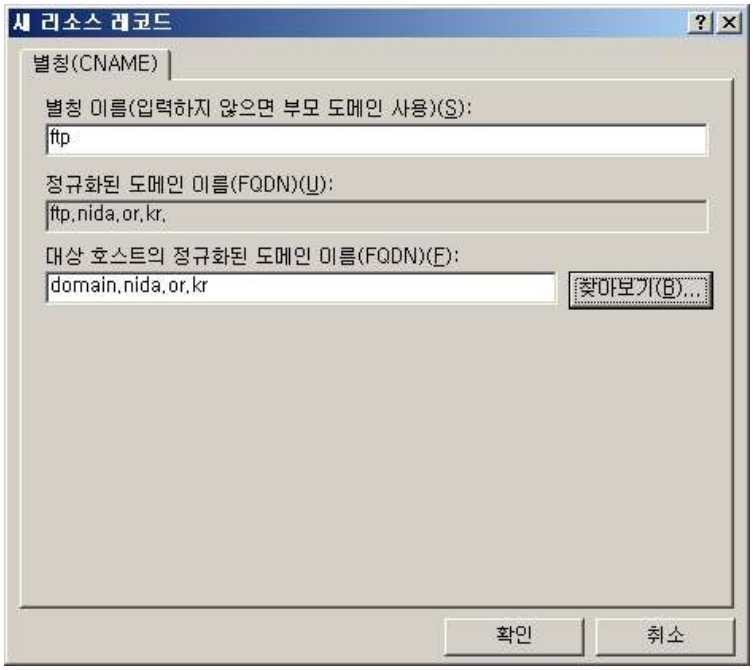
이 영역에 필요한 대로 메일 서버 우선순위를 조정한다. 확인을 클릭하여 영역에 새 레코드를 추가한다. 메일 서버 우선순위는 10이 기본값이며, 메일 서버가 여러 대 있다면 우선순위를 별도로 할당해 줄 수 있습니다. 숫자가 낮을수록 우선순위는 높다.

□ 영역에 CNAME(별칭) 리소스 레코드 추가

o DNS 콘솔 트리에서 적용 가능한 정방향 조회 영역을 마우스 오른쪽 단추로 클릭한 다음 새 별칭(CNAME)을 클릭한다.

[그림 3-24] 새 별칭(CNAME)

o 별칭 이름(입력하지 않으면 부모 도메인 사용) 입력란에 별칭 이름을 입력한다.
- 대상 호스트의 정규화된 도메인 이름(FQDN) 입력란에 이 별칭이 사용될 DNS 호스트 컴퓨터의 정규화된 도메인 이름을 입력한다.
- 옵션으로 찾아보기를 클릭하여 DNS 네임스페이스에서 호스트(A) 레코드가 이미 정의되어 있는 이 도메인의 호스트를 검색할 수 있다.
- 확인을 클릭하여 영역에 새 레코드를 추가한다.


새 리소스 레코드
별칭(CNAME)
별칭 이름(입력하지 않으면 부모 도메인 사용)(S):
ftp
정규화된 도메인 이름(FQDN)(U):
ftp.nida.or.kr.
대상 호스트의 정규화된 도메인 이름(FQDN)(F):
domain.nida.or.kr
찾아보기(B)...
확인
취소

[그림 3-25] 별칭 설정

4장 Q&A

4.1 BIND 관련 Q&A

□ BIND 버전 선택 기준은?

현재 주로 이용되고 있는 BIND 버전은 8.X와 9.X가 있다. BIND 9.X는 기존의 BIND 8.X의 자료구조나 전반적인 시스템 구조를 새로 작성한 것이다. BIND 9.2.1 이전의 BIND 9.X는 BIND 8.X에 비해 제공하는 기능이 떨어졌으나 그 이후는 거의 대동소이하다. 오히려 IPv6 지원, Dynamic Update, DNSSEC, DNS 보안, 그리고 응용 기능들은 BIND 9.X가 충실하다. 일반적으로 (초당 질의수가 수천에 이르는) 아주 바쁜 네임서버는 BIND 8.X를 이용하고, 그 외의 경우에는 최신 버전의 BIND 9.X를 사용하고 있다.

몇몇 ISP에서는 고객편의를 위해 자사 캐시네임서버의 네임서버소프트웨어로 BIND 8 계열을 사용하고 있다. 앞서 말한 것처럼 BIND 8 계열은 성능은 좋으나 DNS 보안상 취약점이 존재한다. 이에, ISP 업체에서도 이를 인식하여 상위버전으로의 업그레이드를 시도하고 있으나 고객들과의 마찰이 있어 쉽게 업그레이드를 하지 못하는 상황이다. 이것은 안전한 DNS 서비스를 위해서 향후, ISP 사업자뿐만 아니라 우리나라 인터넷을 이끌어가는 NIDA가 함께 풀어 나가야할 과제이다.

□ 여러 버전의 BIND가 한 서버에 설치되었을 경우 주의사항?

테스트 또는 버전업그레이드를 위해 한 서버에 여러 버전의 BIND가 설치할 경우, BIND 버전별로 디렉토리(directory)를 생성하여 설치하고 BIND 실행 시 전체경로를 명시하여 수행하기를 권고한다. 그렇지 않고 ./named 형태로 BIND를 실행시켰을 경우, 시스템의 process 상태정보 파악 시 어떤 named 데몬이 실행중인지 알아보기 어렵다. 다음은 BIND 8.X 에서 Debugging 레벨 3을 주어 실행한 예이다.

※ 예) /usr/local/bind/bind8/named -u named -g named -d 3

□ BIND 에서 IPv6가 지원되나?

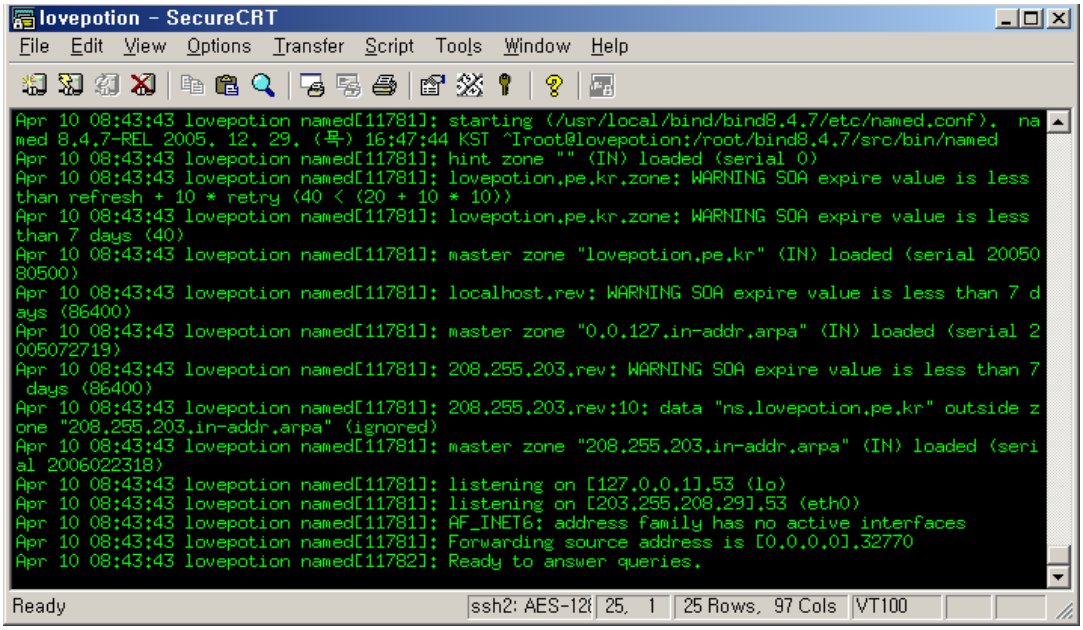
BIND 9에서는 named.conf상의 옵션으로 BIND가 IPv6질의를 인식하도록 하고 있다. 옵션은 다음과 같다. 그러나 BIND 8에서는 네임서버 벤더(Vendor) OS에 따라 지원여부가 다르다. 참고로 IBM의 AIX 5.2/5.3의 경우 BIND 8에서 IPv6는 지원되지 않지만, Sun Solaris 9의 경우에는 BIND 8에서도 IPv6를 지원한다.

```
options {
        listen-on-v6 { any; };                  // 기본포트 53번의 IPv6 NIC를 청취하도록 함
        listen-on-v6 port 1053 { any; };        // 특정 포트 1053번만 청취하도록 설정함
        [...];
}
```

[표 4-1] BIND 9에서 IPv6 지원 옵션

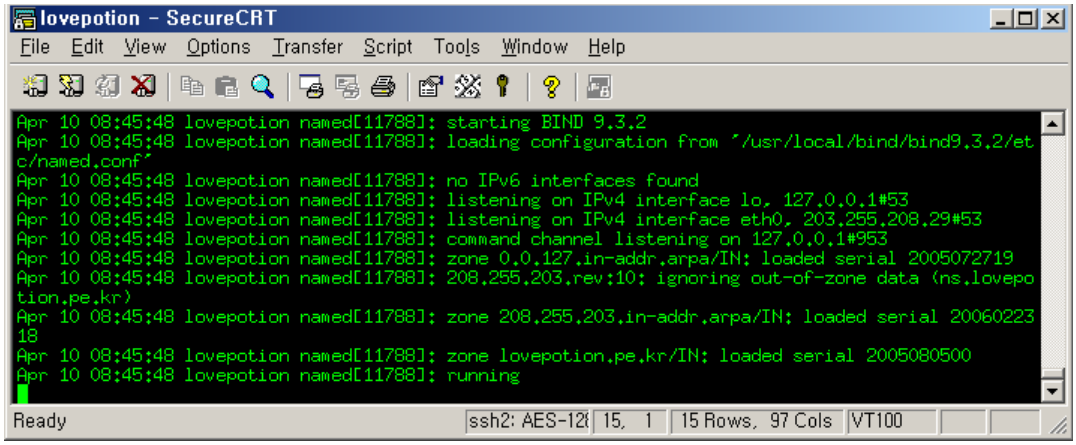
□ BIND 버전 확인 방법은?

일반적으로 BIND가 시작되면 버전에 관련된 정보가 syslog에 전달된다. 따라서 syslog에 저장된 메시지를 검토하여 확인할 수 있다. syslog로 전달된 메시지는 일반적으로 /var/adm/messages 또는 /var/log/messages에 있다. 이 파일 내에 있는 내용을 검색하기 위해 vi와 같은 에디터를 이용할 수도 있다. grep이나 sed, awk와 같은 정규식을 이용할 수 있는 유틸리티를 사용할 수도 있다. 다음 그림은 BIND 8.4.7의 시작 메시지 부분이다. 첫줄에서 시작한 시각과 버전을 확인할 수 있다.



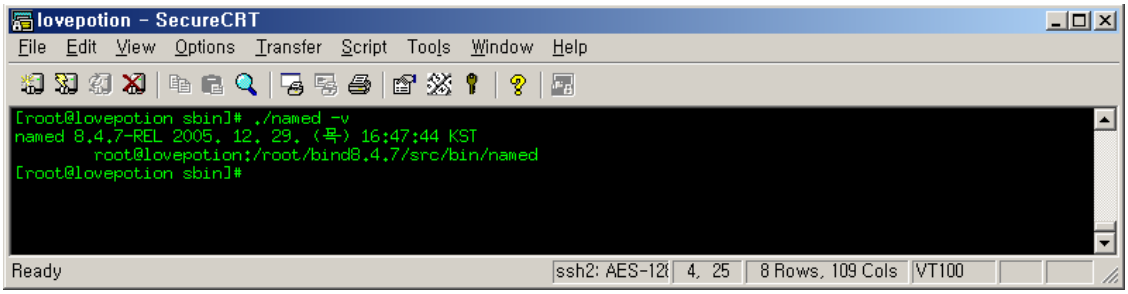
[그림 4-1] named 시작과 관련된 syslog 메시지(BIND 8.X)

다음 그림은 BIND 9.X의 시작과 관련된 syslog 메시지이다. 화면에서 버전정보를 확인할 수 있다. 이외에도 설정파일이나 인터페이스에 대한 정보도 같이 알 수 있다.

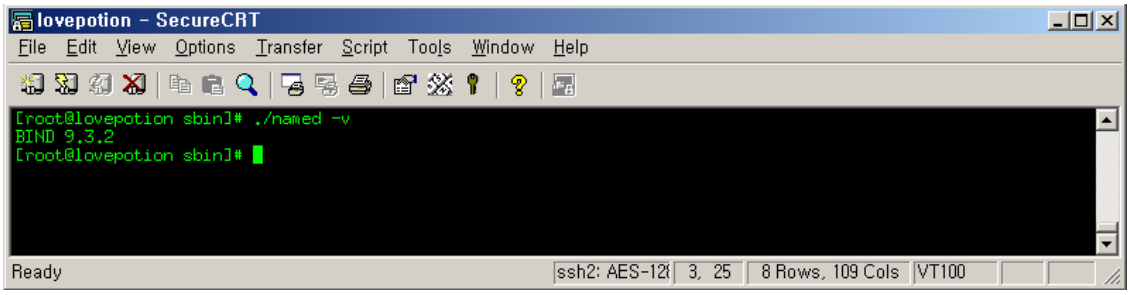


[그림 4-2] named 시작과 관련된 syslog 메시지(BIND 9.X)

또한 named를 실행할 때 -v 옵션을 주어서 화면에 버전을 출력하게 할 수도 있다. 다음은 BIND 8.4.7와 BIND 9.3.2에서 명령을 수행한 화면이다.



[그림 4-3] named -v 로 버전을 확인(BIND 8.X)



[그림 4-4] named -v 로 버전을 확인(BIND 9.X)

참고로, BIND 버전별 취약점 공격을 대비하여 버전정보는 안보이게 하는 것이 좋다. 버전정보를 보이지 않도록 하기 위해서는 BIND 8.2이상 버전에서는 아래와 같이 named.conf를 설정함으로써 BIND 버전 정보 유출을 막을 수 있다.

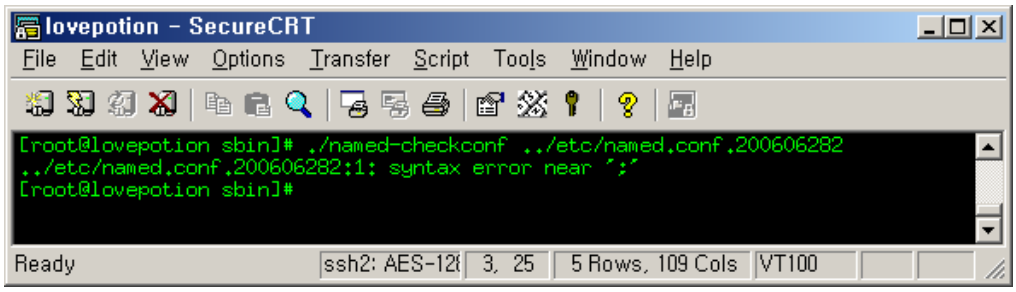
```
options {
        version "None of your business..";
        [...];
}
```

[표 4-2] BIND 8.2 이상에서 버전 정보 설정

□ BIND 에서 named.conf 파일의 설정이 정상인지 확인하려면?

우선 가능한 방법은 named를 실행한 다음 syslog를 살펴서 부팅 과정에서 나온 메시지를 확인한다. 이를 직접 로그 파일을 검색해서 확인할 수도 있고, BIND 9.X인 경우 named -g로 메시지를 출력하도록 할 수도 있다. 오류가 있는 경우 줄 번호와 에러 메시지가 나온다.

named-checkconf 프로그램을 이용하면 named.conf 파일의 설정을 확인할 수 있다. 문법적인 오류가 있는 경우 줄 번호와 에러 메시지를 화면에 출력한다. 다음은 named-checkconf를 수행한 예이다.



[그림 4-5] named-checkconf 실행 화면

이와 마찬가지로 named-checkzone 프로그램을 이용하면 각 영역 파일들의 문법을 확인할 수 있다.

□ BIND 8.X에서 ndc로 네임서버를 제어하고자 하면?

ndc를 인자없이 실행하면 ndc는 /var/run/ndc라는 유닉스 도메인 소켓을 통해 로컬 호스트에서 동작중인 네임서버와 대화하려고 한다. 일부 운영체제는 다른 경로명에 있는 소켓을 이용하기도 하는데 이 소켓은 일반적으로 루트 소유이며, 소유자만 읽고 쓰는 것이 가능하다. BIND 8.2 및 이후 버전의 네임서버는 시작 시에 이 유닉스 도메인 소켓을 생성한다. 소켓의 경로명이나 허가는 controls 구문을 이용해 바꿀 수 있다. 이 유닉스 도메인 소켓은 네임서버를 제어할 권한이 있는 사용자만 접근할 수 있도록 해야 한다.

ndc는 TCP 소켓을 통해 로컬이 아닌 원격 호스트의 네임서버로 메시지를 보낼 수도 있다. 이 기능을 이용하려면 다음의 예처럼 -c 옵션을 주고 원격 네임서버의 이름이나 주소와 제어메시지를 청취(listen)할 포트 번호를 (/)로 구별하여 옵션의 인자로 주면 된다.

```
#ndc -c 192.250.250.13/953
```

[표 4-3] TCP 소켓을 통한 원격 네임서버 제어

이렇게 ndc를 통해 원격 네임서버를 제어하려면, 제어될 네임서버에서도 설정이 필요한데, 제어될 네임서버에서는 다음과 같이 control 구문을 이용하여 특정 TCP 포트를 통해 제어메시지를 기다리도록 설정할 수 있다.

```
controls {
        inet    *    port 953    allow { ndchost; };
};
```

[표 4-4] 네임서버에서의 ndc를 통한 원격 제어를 위한 설정

□ BIND 9 에서 rndc로 네임서버를 제어하려면?

rndc 도 BIND 8.X의 ndc와 같이 네임서버를 제어하는 기능을 하는데, ndc와 비교하여 가장 중요한 변화는 ndc의 경우는 유닉스 도메인 소켓을 이용했는데, rndc는 유닉스 도메인 소켓이 아닌, inet 도메인 소켓을 이용한다는 것이다. 이러한 inet 도메인 소켓의 사용은 원격 네임서버의 제어도 가능하게 한다. 그리고 보안 강화를 목적으로 반드시 키 값을 요구한다. 따라서 서로간의 통신에 이용할 비밀키를 설정하는 것이 필요하며 이러한 설정을 위한 rndc.conf 파일을 요구한다.

named.conf 와 rndc.conf 파일 생성을 돕는 명령어가 제공되는데, 그것을 rndc-confgen이다. 또한 BIND의 버전이 9.2 이후라면 rndc-confgen -a를 이용하여 rndc와 named가 공통으로 이용할 키 파일을 생성하도록 할 수 있다. 이 때 named.conf 내에 controls 문장이 없어야 하고, rndc.conf 파일은 존재하면 안 된다. 둘 중 하나라도 있으면 자동 설정이 되지 않는다.

□ BIND 9에서 rndc로 네임서버를 여러 개 제어하려면?

네임서버를 여러 개 제어할 것이라면 각각의 네임서버에 서로 다른 키를 연계시킬 수 있다. 이땐 각 키를 별도의 key 구문으로 정의하고 서로 다른

server 구문에 연계하면 된다. 그런 후 rndc에 -s 옵션으로 제어할 서버를 지정해 실행한다. 또한 아직 특정한 네임서버에 연계시키지 않은 키가 있다면 명령 행에 -y 옵션을 이용해 특정키를 지정할 수 있다. 네임서버가 제어 메시지를 위해 953번이 아닌 다른 포트를 청취하는 상황이라면 -p 옵션을 이용해 어떤 포트로 연결해야 할지 rndc에게 알려주어야 한다.
제어 대상이 되는 네임서버에서는 설명한 바와 같이 rndc-confgen을 통해 만들어진 key 문장과 controls 문장을 named.conf에 추가하고, 특히 제어할 rndc의 주소를 등록한다.

```
controls {
        inet     *   allow { 127.0.0.1; 192.250.250.3; }  keys {"rndc-key"};
};
```

[표 4-5] 네임서버에서의 rndc를 통한 원격 제어 설정

□ 네임서버의 재시작 없이 영역데이터를 갱신하고자 하면?

rndc reload 영역이름 또는 ndc reload 영역이름으로 네임서버의 재시작 없이 영역 데이터를 변경할 수 있다.

□ 네임서버의 재시작이나 리로드(reload) 없이 영역을 추가하거나 삭제하고자 하면?

named.conf 파일에 새로운 영역을 추가하거나 기존 영역을 삭제한 후,
rndc reconfig 또는 ndc reconfig를 하면 추가되거나 삭제된다.

□ 영역 데이터 전송을 바로 시작하려면?

슬레이브 네임서버에게 영역 데이터 전송을 바로 시작하도록 하려면 rndc refresh 영역이름 또는 ndc reload 영역이름을 하면 된다.

□ 전에 실행한 것과 같은 인수로 네임서버를 재시작하려면?

rndc에서는 재시작하는 기능이 아직 제공되지 않고 있다. 그러나 ndc에서는 ndc restart로 재시작이 가능할 뿐만 아니라 ndc exec로 바로 전에 실행했던 인수를 그대로 받아 재시작할 수 있다.

□ 네임서버의 캐시를 저장하고자 하면?

rndc dumpdb 또는 ndc dumpdb를 이용하여 캐시된 내용을 디스크로 저장할 수 있다.

□ 네임서버의 캐시를 지우고자 하면?

BIND 9.X인 경우 rndc flush로 캐시를 지울 수 있다. 그러나 BIND 8.X의 경우 네임서버를 다시 시작하여한다.

□ 네임서버를 영역의 주 네임서버로 설정하는 방법은?

영역의 주 네임서버로 설정하고자 하는 경우 named.conf 파일에 다음과 같이 type 속성 값을 master로 설정하면 된다. 다음은 named.conf의 내용 중에서 nida.or.kr 의 주 네임서버로 설정하는 부분이다.

```
zone "example.co.kr"  IN  {
        type master;
        file "db.example.co.kr";
};
```

[표 4-6] named.conf 중 주 네임서버로 설정 부분

□ 네임서버를 영역의 슬레이브(slave) 네임서버로 설정하는 방법은?

영역의 슬레이브 네임서버로 설정하고자 하는 경우 named.conf 파일에 다음과 같이 type 속성 값을 slave로 설정하고 주 네임서버들을 지정하면 된다. 다음은 named.conf의 내용 중에서 nida.or.kr.의 슬레이브 네임서버로 설정하는 부분이다.

```
zone "example.co.kr"  IN  {
        type slave;
        master { 192.250.250.11};
        file "bak.example.co.kr";
};
```

[표 4-7] named.conf 중 슬레이브 네임서버 설정 부분

□ 네임서버가 여러 영역을 담당하도록 설정하는 방법은?

하나의 네임서버가 여러 영역을 담당하도록 하기 위해서는 named.conf 파일에 추가된 영역 당 존 문장을 삽입하면 된다. 이때 네임서버의 type은 master 일수도 있고 slave 일수도 있다.

```
zone "example.co.kr"  IN  {
        type master;
        file "db.example.co.kr";
};

zone "example2.co.kr"  IN  {
        type master;
        file "db.example2.co.kr";
};
```

[표 4-8] named.conf 중 복수 영역을 담당하도록 설정하는 부분

□ 하나의 영역을 여러 파일로 나누어 담당하려면?

영역을 여러 파일로 나누어 관리하고자 하면 영역 파일 이름에 명시된 파일 내에 $include 문장으로 다른 파일을 포함시키면 된다.

□ Dynamic Update를 허용하려면?

일반적으로 다음과 같이 allow-update 문장을 named.conf 파일에 포함하면 된다.

```
zone "example.co.kr"  IN  {
        type "db.example.co.kr"`;
        allow-update { 192.250.250.12; };
};
```

[표 4-9] BIND 8.X 주 네임서버에서의 Dynamic UPDATE 허용 설정

allow-update 문장을 통한 dynamic update 허용은 보안상 문제가 있으므로 지원이 가능하면 반드시 update-policy 문장을 이용하여 TSIG를 이용한 갱신을 이용하는 것이 좋다. BIND 9.X의 경우 TSIG-signed UPDATE을 지원하는데, 이때에는 update-policy 라는 문장을 추가로 필요로 한다.

```
update-policy {
          grant|deny keyname nametype domain-name [type[...]];
          [...];
};
```

[표 4-10] update-policy 문

여기서 keyname은 dynamic update에서 이용할 TSIG 키 이름이고, nametype은

name, submain, wildcard, self 형태로 나타날 수 있다. 부가적으로 나타날 수 있는 type은 A, MX, NS와 같은 레코드 타입을 의미한다.

예를 들면, update-key라는 이름의 키를 이용하면서, 도메인이름과 같은 경우만 www.nida.or.kr의 A 리소스 레코드의 갱신을 허용하고자 하는 경우 다음과 같이 설정한다.

```
zone "example.co.kr"  IN  {
        type "db.example.co.kr";
        update-policy { grant update-key name www.example.co.kr  A; };
};
```

[표 4-11] update-policy 설정 예

□ 슬레이브 네임서버가 주 네임서버에게 Dynamic Update를 전달하도록 하려면?

BIND 9.X의 경우 슬레이브 네임서버가 주 네임서버에게 dynamic update를 전달할 수 있다. 이를 위해서 슬레이브 네임서버의 named.conf 파일의 zone 문장에 allow-update-forwarding 문장을 추가한다.

```
zone "example.co.kr"  IN  {
        type slave;
        master { 192.250.250.1; };
        file "bak.example.co.kr";
        allow-update-forwarding { localhosts; };
};
```

[표 4-12] allow-update-forwarding 설정 예

□ NOTIFY 메시지를 제한하려면?

NOTIFY 메시지를 제한하는 대표적인 경우가 슬레이브 네임서버가 동일한 메시지를 또 보내는 것을 막고 싶을 때이다. 원래는 주 네임서버가 NOTIFY를 보내면 이를 받은 슬레이브 네임서버가 다른 네임서버들에게 또 NOTIFY 메시지를 보내도록 되어 있다. 다음은 슬레이브 네임서버에서 NOTIFY를 보내지 않도록 설정한 예이다.

```
zone "example.co.kr"  IN  {
        type slave;
        master { 192.250.250.1; };
        file "bak.example.co.kr";
        notify no;
};
```

[표 4-13] 슬레이브 네임서버에서 NOTIFY off

또한 명시된 네임서버들에게만 NOTIFY를 보내도록 설정할 수 있다. 이때에는 notify를 explicit로 설정하고, also-notify 문장에서 보낼 네임서버들을 나열한다.

```
zone "example.co.kr"  IN  {
        type slave;
        master { 192.250.250.1; };
        file "bak.example.co.kr";
        notify explicit;
        also-notify { 192.250.250.1; };
};
```

[표 4-14] 명시된 네임서버에게만 NOTIFY 전송 예

□ IXFR을 설정하려면?

BIND 9.X의 경우 IXFR이 디폴트로 설정되어 있으므로 별도의 설정이 필요 없다. 그러나 BIND 8.X의 경우 우선 슬레이브 네임서버를 다음과 같이 설정하여야 한다.

```
server 192.250.250.11 {
        support-ixfr yes;
};
```

[표 4-15] BIND 8.X 슬레이브 네임서버에서 IXFR 설정

또한 주 네임서버에서도 IXFR 요청을 받을 수 있도록 다음과 같이 설정한다.

```
options {
        directory   "/var/named";
        maintain-ixfr-base    yes;
};
```

[표 4-16] BIND 8.X 주 네임서버에서 IXFR 설정

□ 네임서버가 사용이 허가된 IP만 질의하게 하려면?

BIND는 내부와 외부에서 구분하여 사용을 제한할 수 있도록 설정할 수 있다. 주요 네트워크로부터 질의에 대해서만 응답할 수 있도록 제한하기 위해서 사용한다. named.conf에 다음과 같이 질의를 허용할 IP 블록을 설정한다.

```
options {
        directory   "/dns1"
        allow-query { 200.1.11; 200.1.2.0/24 };
};
```

[표 4-17] 허가된 IP로만 질의하게끔 하는 named.conf 설정

설정완료 후 네임데몬에 설정을 반영한다.

```
#rndc reconfig
```

[표 4-18] 설정을 네임데몬에 반영

□ 질문자에 따라 응답을 달리하려면?

네임서버가 동일한 질문을 질문자에 따라 달리 응답을 하도록 하고자 하는 경우가 있다. 대표적으로 망 내부 호스트에서 온 질의와 망 외부에서 온 질의를 달리 처리하고자 하는 경우가 있을 수 있다. 이 때 BIND 9.X의 뷰를 이용한다. 이를 위해 먼저 acl 문장을 이용하여 질의자들을 구분 짓고, 두 개의 view 문장으로 각기 다른 파일을 참조하여 응답을 하도록 설정한다.

```
acl     internal     { 192.250.250/24; };

view internal {
              match-client { internal; };
              zone "example.co.kr"  IN  {
                    type master;
                    file "db.example.co.kr.internal";
              };
};

view external {
              match-client { any; };
              zone "example.co.kr"  IN  {
                    type master;
                    file "db.example.co.kr.external";
              };
};
```

[표 4-19] 뷰(View) 설정 예

□ 시리얼 넘버를 조정하려면?

거대 도메인을 관리하는 매니저들의 실수 중 하나는 잦은 업데이트작업으로 인한 잘못된 Serial 넘버링이다. DNS 시리얼 넘버는 32비트의 부호 없는 정수 범위로 설정가능하다. 일반적으로 2006050101로 날짜와 그날 업데이트한 횟수로 표기한다. 하지만 19990205010과 같이 실수로 삽입된 '0'은 해당 필드를 오버플로우 시킨다. 따라서 Secondary의 Zone은 장기간 업데이트 되지 않을 수 있다. 다음과 같이 문제를 해결할 수 있다.

- o Secondary를 직접 관리한다면, 먼저 Primary Zone의 Serial을 정상적으로 조정한다. Secondary에 저장되어 있는 Zone 파일(Zone Transfer된)을 삭제한 후 BIND를 재 구동한다.
- o BIND 4.8.1보다 최신이면서 BIND 9보다는 오래된 버전이라면 시리얼 번호를 0으로 설정하면, 타기관의 Secondary 네임서버에 존이 전송된다.
- o BIND 4.9 이후의 DNS 시리얼 번호는 순차 공간 산술법(sequence space arithmetic)을 이용하는데 이는 임의의 시리얼 번호에 대해 번호 공간 내 번호의 절반(2,147,483,647 번호들)이 그 시리얼 번호보다 작으며 나머지 절반은 더 크다는 의미이다.

  예를 들어 현재 영역 시리얼 번호가 25,000인데 번호 1부터 다시 시작하고 싶다면 두 단계를 거쳐 시리얼 번호 공간을 바꿀 수 있다. 우선 가능한 최대의 증가치를 시리얼 번호에 더한다.(25,000+2,147,483,647 = 2,147,508,647) 더한 결과 값이 4,294,976,295(32비트 최대치)보다 크면 그 수에서 4,294,967,296을 빼 번호공간의 처음으로 되돌린다. 시리얼 번호를 바꾼 후에는 모든 슬레이브가 영역의 새로운 데이터를 가져갈 때까지 기다려야 한다. 그런 다음 영역 시리얼 번호를 원하는 값인 1로 바꾸면 현재 시리얼 번호(2,147,508,647)보다 크다. 슬레이브들이 새로운 영역 데이터를 가져간 후에는 원하는 일이 끝난 것이다.

□ BIND 버전별 기능 호환 조건들은 어떻게 되나요?

버전별 기능 호환은 아래와 같다. (자세한 사항은 DNS&BIND 참고)

기능		BIND 버전			
		4.9.7	8.1.2	8.2.3	9.1.0
다중프로세서지원					○
동적업데이트(Dynamic Update)			○	○	○
재귀(recursive) 기능 비활성화		○	○	○	○
재귀(recursive) 액세스 리스트		○	○		
NOTIFY			○	○	○
IXFR 영역전송				○	○
TSIG서명된 동적업데이트				○	○
TSIG기반 업데이트 정책					○
뷰(VIEW)					○
라운드로빈(Round Robin)		○	○	○	○
포워딩(Forwarding)		○	○	○	○
포워드 영역 (Forward zone)				○	○
포워더에 대해 RTT이용				○	
설정 가능한 RRset 순서				○	
설정 가능한 Sort list		○		○	○
DNSSEC 지원	SIG,KEY,NXT RR들 로딩			○	○
	'Trust chain' 확인				○
	안전 영역의 동적업데이트				○
IPv6 지원	AAAA 레코드	○	○	○	○
	A6 레코드				○
	DNAME 레코드				○
	비트문자열 레이블				○
	DNAME과 A6 체인 따름			○	
	질의 Access list	○	○	○	○

[표 4-20] BIND 버전별 기능 호환 조건표

□ TSIG를 통해 동적업데이트 또는 존 전송을 사용할 때 TSIG설정을 정확히 했음에도 해당 서버에서 TSIG를 reject할 경우?

이 경우 client와 server간의 시간차이가 발생했을 확률이 높다. 두 시스템간의 시간동기화(ex, ntp 프로토콜 이용)를 시킨 후, 재전송하기를 권장한다.

□ BIND 버전별 취약점 정보는 어디서 볼 수 있나?

BIND 버전별 취약점 정보는 http://www.isc.org/products/BIND/bind-security.html를 방문하면 자세히 볼 수 있다. BIND 버전별 취약점 공격을 예방하기 위해서는 이미 앞서 말한바와 같이 namd.conf 파일의 options 부분을 이용하여 BIND 버전 정보를 숨기는 방법이 있고, 또 한 가지 방법은 최신의 BIND 버전으로 업그레이드 하는 것이다.

4.2 네임서버 관련 Q&A

□ 자주 가던 홈페이지 접속이 안 될 경우?

자주 가던 홈페이지에 접속이 안 될 경우 우선 기본적인 사항들부터 점검을 해본다. 네트워크 회선을 체크하고, DHCP(Dynamic Host Configuration Protocol)가 아닐 경우 설정된 로컬(local) 네임서버 정보 및 설정상태를 확인한다. 로컬네임서버의 상태 확인결과 이상이 없으면 해당도메인이 현재 운영중인지를 점검한다. 도메인의 운영정보는 http://whois.nida.or.kr을 참조한다. 도메인이 운영중인데도 접속이 안 될 경우 해당 서버가 장애일 가능성이 높다.

□ 네임서버 설정 상태 확인하는 방법은?

해당 도메인이 사용하고 있는 네임서버가 제대로 동작하는지를 점검하기 위해서 dig나 nslookup 같은 툴(tool)을 사용할 수 있다. 그러나 더 편하게 점검하고 싶다면 진흥원에서 제공하는 네임서버 점검 웹페이지(http://han1.nic.or.kr/dnschk/)를 이용하기를 권장한다. 도메인 입력란에 해당 도메인만 입력하여 실행시키면, 간단히 도메인의 네임서버 상태 점검할 수 있다.

□ Lame Delegation은 무엇인가?

Lame delegation이란 Namespace 상에서 깨어진 링크(Link)를 말한다.

```
example.co.kr.      IN NS ns1.example.co.kr.
                    IN NS ns2.example.co.kr.
```

[표 4-21] Lame Delegation 예

예를 들어 example.co.kr 이 위와 같이 두 개의 네임서버를 갖지만, 두 서버 중 하나 혹은 모두가 해당 도메인에 대한 Authority를 갖지 않는 경우, 즉 Primary, Secondary 설정이 안 되어 있을 경우가 Lame delegation에 해당된다.

□ 네임서버 선택은 어떻게 이루어지는가?

네임서버 간에 질의, 응답에 소요되는 시간을 Round Trip Time이라 한다. (Recursive 모드에서 총 검색 시간이 아니다) BIND는 내부적으로 타 네임서버에 대한 RTT 값을 기록하고 있다가, 요청 도메인에 대한 다수의 Authority NS 중 RTT 값이 가장 낮은 네임서버로 먼저 질의한다. Authority NS들에 대한 RTT 정보를 갖고 있지 않을 경우엔, 해당 네임서버 전체에 질의(동시에)를 보내어

빠른 응답을 얻음과 함께 부가적으로 RTT를 측정한다. RTT가 측정된 다음부터는 해당 도메인에 대한 요청이 RTT가 가장 적은 서버로 먼저 보내어 진다. 또한, 몇몇 서버만이 계속 사용되는 문제를 막기 위해 질의를 전송할 때마다 해당 네임서버에 대한 RTT값을 조금씩 증가시킨다.

이와 관련하여, 해당도메인 사용자가 자신의 네임서버를 1차(Master), 2차(Slave) 형태로 운영할 경우 외부 질의는 1차 네임서버에 질의를 요청한 후 응답이 없으면 2차 네임서버로 질의 요청을 하는 것이 아니라 미리 기록된 RTT값을 참고하여 응답이 빠른 서버쪽으로 질의를 요청하게 된다.

□ 네임서버는 반드시 복수개로 운영해야 하는가?

하나의 네임서버로 영역을 담당하는 것도 가능하지만 안정성을 이유로 둘 이상의 네임서버를 운영하는 것이 권고되고 있다. 그 중 하나는 주 네임서버로, 나머지는 슬레이브 서버로 설정하는 것이 일반적이다.

□ 네임서버에 새로운 호스트를 추가하려면?

새로운 호스트가 들어오면 영역 파일에 A 리소스 레코드와 PTR 리소스 레코드가 추가되어야 한다. A 리소스 레코드는 db.nida.or.kr 파일에 추가되어야 하고, PTR 리소스 레코드는 250.250.192.in-addr.arpa 파일에 추가되어야 한다.

```
newhost.example.co.kr.        IN      A       192.250.250.151
```

[표 4-22] db.example.co.kr에 새로운 A 레코드 추가 부분

```
151.250.250.192.in-addr.arpa        IN      PTR     newhost.example.co.kr.
```

[표 4-23] 250.250.192.in-addr.arpa에 새로운 PTR 레코드 추가부분

□ 네임서버에 기존 호스트의 이름(별명)을 추가하려면?

이미 등록되어 있는 호스트에게 새로운 이름(별명)을 부여할 경우가 있을 수 있다. 예를 들면, 이미 운영되고 있는 웹 서버를 새로 ftp 서버로도 동작하게 하는 경우 www.nida.or.kr 보다는 ftp.nida.or.kr 이라는 이름이 더 적합할 것이다. 이런 경우 CNAME 리소스 레코드를 이용한다.

```
ftp.example.co.kr.        IN      CNAME       ns1.example.co.kr.
```

[표 4-24] db.example.co.kr에 새로운 CNAME 레코드 추가 부분

□ 네임서버에 메일 호스트를 추가하려면?

일반 호스트와는 달리 메일 호스트는 따로 MX 리소스 레코드를 이용하여 추가한다. 다음과 같이 MX 리소스 레코드를 추가하는 경우 user@nida.or.kr로 보내는 메일을 mail.nida.or.kr 서버가 받아 처리할 수 있다. MX 리소스 레코드에는 preference 필드가 있는데, 여기에는 0부터 65535까지의 정수 값을 설정할 수 있다. 이 값 자체가 중요한 것은 아니고, 메일 서버가 여러 개 존재할 때 어떤 메일서버를 우선적으로 이용할 것인지를 나타낸다. 값이 낮은 것이 더 높은 우선순위를 갖는다.

```
example.co.kr.      IN    MX    0     mail.example.co.kr.
example.co.kr.      IN    MX    10    smtp.example.co.kr.
```

[표 4-25] db.nida.or.kr에 MX 레코드 추가 부분

□ www 없이 도메인 이름으로 바로 웹 서버에 연결되도록 하려면?

일반적으로 www.example.co.kr 이라고 입력하여 웹서버에 접속하지만, 단지 도메인이름 nida.or.kr만 입력해도 웹서버에 접속하도록 할 수도 있다. 이렇게 하고자 하면 A 리소스 레코드의 첫 필드에 호스트 이름이 아닌 영역 이름을 입력하고 웹 서버의 IP 주소를 입력하면 가능하다. 웹서버가 여러 개 존재하면 그만큼의 A 리소스 레코드를 더 설정하면 된다.

```
example.co.kr.      IN    A     192.250.250.1
example.co.kr.      IN    A     192.250.250.2
```

[표 4-26] 영역 이름으로 바로 웹서버에 접속하도록 A 추가

□ 복수 개의 웹서버(web server)로 부하를 분산하려면?

복수 개의 웹서버를 운영하면서 전체 부하를 분산하고자 하는 경우가 있을 수 있다. 이 때 영역 파일에 같은 이름을 갖는 A 리소스 레코드를 복수 개 설정하면 라운드-로빈(Round-Robin) 방식으로 서비스를 제공하게 된다. 다음 예는 3개의 웹서버가 하나의 영역을 담당하는 경우를 보여준다.

www.example.co.kr.	IN	A	192.250.250.1
www.example.co.kr.	IN	A	192.250.250.2
www.example.co.kr.	IN	A	192.250.250.3

[표 4-27] 라운드-로빈 방식으로 부하분산

□ 멀티홈 호스트를 등록하려면?

두개의 인터페이스를 가져서 두개의 망에 속한 호스트를 멀티홈(Multihomed) 호스트라고 한다. 대부분 이런 호스트는 라우팅이나 파이어월 기능을 하는 경우가 많다. 이런 호스트를 등록하고자 하면 영역 파일에 두개의 A 리소스 레코드를 추가하고, 각각의 리버스 영역 파일에 PTR 리소스 레코드를 추가하면 된다.

abc.example.co.kr.	IN	A	10.0.0.1
abc.example.co.kr.	IN	A	192.250.250.5

[표 4-28] db.nida.or.kr 에 멀티홈 호스트 추가 부분

1.0.0.10.in-addr.arpa.	IN	PTR	abc.example.co.kr.

[표 4-29] 0.0.10.in-addr.arpa에 멀티홈 호스트 추가 부분

1.0.0.10.in-addr.arpa.	IN	PTR	abc.example.co.kr.

[표 4-30] 250.250.192.in-addr.arpa에 멀티홈 호스트 추가 부분

□ 호스트의 주소를 변경하고자 하면?

호스트의 주소를 바꾸고자 할 때 바로 영역 파일을 변경하는 것은 문제가 발생할 수 있다. 일반적으로 네임서버에서는 질의의 응답으로 받은 정보를 캐시에 보관하는데 이 때의 유효 시간이 TTL 값이다. 따라서 먼저, 일반적으로 하루 정도로 설정되어 있는 해당 A 리소스 레코드와 PTR 리소스 레코드의 TTL 값을 60초와 같은 낮은 값으로 먼저 변경하고, 충분한 시간이 흐른 후 변경 작업을 수행하고, 그 다음 TTL 값을 높여주는 것이 좋다.

□ 네임서버 TTL, SOA 레코드 값은 어느 정도가 적당한가?

네임서버 존의 TTL값과 SOA 레코드의 Refresh, Retry, Expire 시간 값들은 초(second) 단위로 설정되어 있다. 그러나 이런 값들은 특정 값으로 정의되어 있지 않고, 네임서버 관리자가 자신의 사이트에 맞게 적절히 설정하는 사용하고 있다. 이와 관련하여, NIDA에서는 네임서버 담당자가 초기에 각 필드 값

설정 시 어느 정도 도움이 되도록 각 필드 값들에 대해 아래와 같이 정의하니 설정 시 참고하기 바란다. 아래의 설정은 권장사항으로 각 사이트의 상황에 따라 적절히 수정하여 사용할 수 있다.

```
$TTL    86000    ; 1일
example.co.kr.   IN  SOA  ns1.example.co.kr.  root.example.co.kr.  (
                                2006080100        ; 시리얼번호
                                10800             ; 3시간 후 리플래시
                                3600              ; 1시간 후 재시도
                                604800            ; 1주일 후 만료
                                3600 )            ; 1시간의 부정적 캐싱 TTL
```

[표 4-31] TTL 및 SOA 레코드 필드 값 설정 예

□ 호스트의 위치를 DNS에 저장하고자 하면?

영역 파일 내에 호스트의 위치를 저장하고자 할 때 이용할 수 있는 리소스 레코드는 TXT와 LOC이다. 이 중 TXT 리소스 레코드는 호스트의 위치를 포함한 어떠한 설명도 포함할 수 있는 레코드이고, LOC 리소스 레코드의 경우 위도와 경도로만 위치를 표현한다. DNS 운영 초창기에는 DNS에 호스트의 위치, 관리자, 기능 등 다양한 정보를 담는 것이 추세였지만, 보안 등의 이유로 제공하는 정보를 제한하는 것이 현재의 추세이다.

□ Linux Kernel 2.2.x 버전에서 --enable-threads 옵션으로 BIND 9을 build 한 후 "-u" 옵션이 적용되지 않는 이유?

Kernel 2.2.x 버전의 Linux threads는 완전히 POSIX threads(pthreads) 표준을 지원하지 않는다. 특히, setuid() 함수는 전체 process가 아닌 오직 현재의 thread에 한에서만 동작한다. 이러한 제한 때문에, Linux 기반의 BIND 9은 setuid()함수를 다른 지원되는 OS 플랫폼처럼 사용할 수 없다. Linux Kernel 2.2.18 or 2.3.99-pre3 에서는 이 기능을 제공하고 있다.

참고로, POSIX(Portable Operating System Interface for Computer Environment)는 유닉스 운영체계에 기반을 두고 있는 일련의 표준 운영체계 인터페이스이다.

□ Linux 에서는 named 프로세서가 5개 이상 보이는 경우가 있다.

Linux 에서 named를 실행시키고 ps 명령을 수행할 경우, named가 여러 개 보이는 것을 목격할 수 있다. named thread 개수는 대략 n+4개 정도이고, 여기서 n은 CPU 개수이다. 유념할 사항은 전체 메모리 사용량은 각

process 별로 누적되는 것이 아니다. 각각의 process는 10MB 정도의 메모리를 사용하고 있다면, 전체 total 메모리소비량 역시 10MB 이다.
상위버전의 Linux 에서는 ps 명령을 수행하면 개별적인 threads들을 보여주지 않는다. 이 경우 전체 named process들을 보기 위해서는 -L 옵션을 사용해야한다.

□ Linux 에서 다음과 같은 Error 메시지가 나오는 이유?

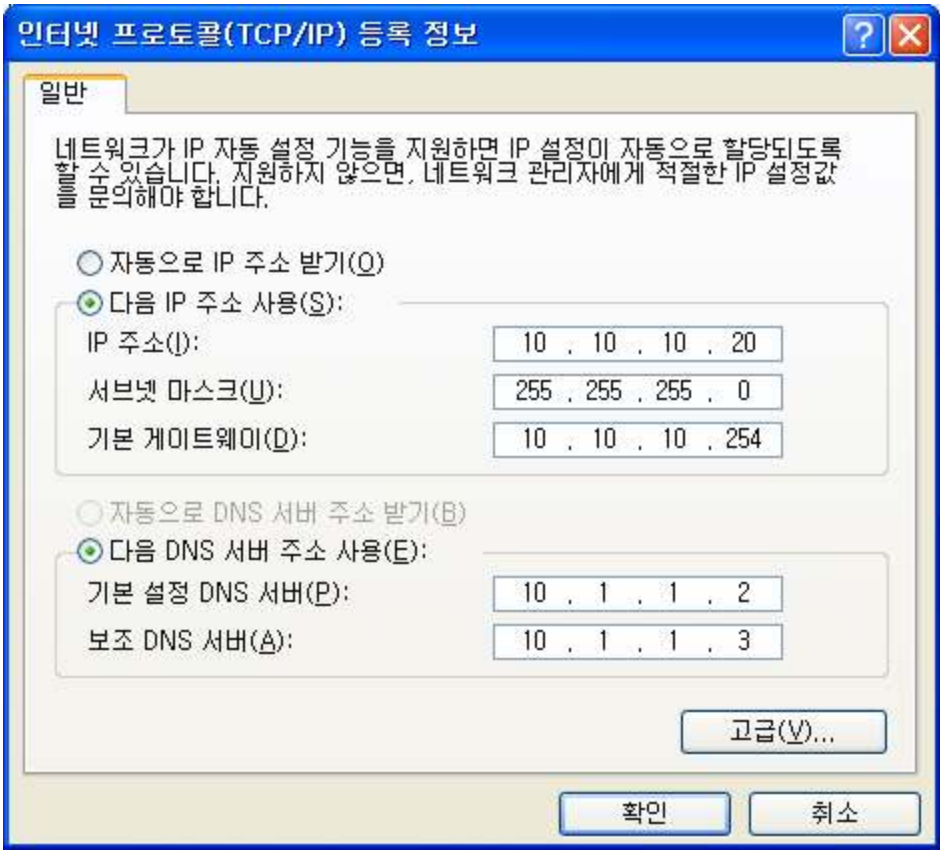
Linux Kernel 버그로 해당 커널 패치 또는 업그레이드를 해야 한다.

```
errors message :
        general: errno2result.c:109: unexpected error:
        general: unable to convert errno to isc_result: 14: Bad address
        client: UDP client handler shutting down due to fatal receive error: unexpected error
```

[표 4-32] 리눅스 커널 에러 메시지

□ 윈도우 서버에서 네임서버 선택은 어떻게 이루어지는가?

Wiindows NT4.0 또는 2000이상에서 리졸버 알고리즘은 다음과 같다. 리졸버는 찾을 DNS 서버 목록의 첫 네임서버에게 첫 질의를 전송하고 1초를 기다린다. 응답이 없으면 질의를 재전송하는데 이때는 자신이 이미 알고 잇는 모든 네임서버에게 다시 보낸다. 만일 어떤 네임 서버도 2초 이내에 응답하지 않으면 리졸버는 네임 서버 모두에게 다시 질의를 재전송한다. 재전송을 다시 할 때마다 시간제한은 계속 2배로 늘어나고, 총 4번의 재전송을 하게 되며 총 15초가 소요된다. 리졸버가 참고하는 DNS 목록은 제어판의 네트워크 설정 상에 나와 있다. [MS기술정보문서 Q198550 참고]



[그림 4-6] 사용자 네트워크 설정 예

□ DNS에서는 별도의 보안 솔루션을 제공하는가?

DNS의 확장으로서 제공되는 DNSSEC(DNS Security Extensions) 기능은 DNS 메시지에 공개키 기반의 전자서명 기능을 제공하여 DNS 데이터에 대한 응답이 신뢰할 수 있는가에 관한 인증 메커니즘을 제공한다. DNSSEC은 보안 기능 제공을 원하는 최상단의 네임서버부터 사용자의 리졸버까지 완벽하게 구현되어야 하며, 이에 따른 고려사항 및 안전한 적용을 위하여 충분한 시험 및 시범적용을 통하여 단계적으로 kr도메인에 적용할 예정이다.

□ DNS 서비스만을 위한 방화벽 정책은?

네임서버는 다른 용도로 사용하지 않고 방화벽에서 네임서비스에 필요한 포트(Port)만 개방함에 따라 캐시오염(Cache Poisoning) 등의 침해의 가능성을 줄일 수 있다. 이에 따라 DNS 만을 위한 별도의 네트워크를 구축하고 DNS 서비스에 필요한 포트만 개방함으로써 보다 안전하게 DNS 서버를 관리할 수 있다.

DNS 서비스에 필요한 포트는 일반적인 질의의 경우 UDP 53과 1024 ~ 65535의 포트를 사용하며, UDP의 메시지가 512Byte 초과 시 또는 Zone Transfer 시에는 TCP 로 질의할 수 있다.

Description	Source IP	Source Port	Destination IP	Destination Port
Queries from clients	Internal network	>1023/udp, >1023/tcp	Name server	53/udp, 53/tcp
Replies to clients	Name server	53/udp, 53/tcp	Internal network	>1023/udp, >1023/tcp
Outbound recursive queries	Name server	53/udp, 53/tcp, >1023/udp, >1023/tcp	Any	53/udp, 53/tcp
Replies to recursive queries	Any	53/udp, 53/tcp	Name server	53/udp, 53/tcp, >1023/udp, >1023/tcp

[표 4-33] 네임서비스를 위한 방화벽 정책 예시

□ 피싱(Phishing)과 파밍(Pharming)의 차이점은?

피싱(Phishing)은 '개인정보를 낚시하듯이 낚는다는 의미'로, 사용자를 속이기 위해 미끼가 되는 가짜 이메일을 보내고 클릭을 유도해, 사용자를 가짜 사이트(예: 인터넷뱅킹)로 유인하여 로그인, 인증번호, 신용카드번호 정보 등과 같은 개인정보를 획득하는 기법이다.

파밍(Pharming)은 합법적으로 소유하고 있던 사용자의 도메인을 탈취하거나 도메인 네임시스템(DNS) 이름을 속여(Spoofing) 사용자들이 진짜 사이트로 오인하도록 유도, 개인정보를 훔치는 피싱의 새로운 수법으로, DNS 캐시오염(Cache Poisoning)과 같이 가짜 데이터를 삽입(Insert)시켜 공격자가 의도한 방향으로 대규모의 피해를 입힐 수 있다는 의미로 Pharming 이 쓰이고 있다.

개인의 인증번호, 신용카드번호, 계좌번호 등 개인의 금융 서비스 관련 정보를 수집하여 활용한다는 공통점 지니나, 파밍은 금융기관의 인터넷 사이트 자체를 위조하여 해당 금융서비스의 마비를 가져올 수 있으며, 파밍이 성공할 경우, 해당 DNS를 이용하는 모든 이용자가 공격에 완전히 노출되므로, 이메일 등 미끼를 클릭해야 공격이 성공하는 피싱과는 비교할 수 없이 파급효과가 크다.