**YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)**

✅ **YANG은 NETCONF에서 사용할 데이터 모델링 언어**  
✅ **NETCONF 프로토콜을 통한 구성(Configuration) 및 상태(State) 데이터 모델링, 원격 프로시저 호출(RPC), 알림(Notification) 지원**

**1️ 문서 개요 (Abstract)**

🔹 **YANG** = **NETCONF에서 사용되는 데이터 모델링 언어**  
🔹 지원 기능:

* **구성 데이터 (Configuration Data) 모델링**
* **상태 데이터 (State Data) 모델링**
* **NETCONF RPC (Remote Procedure Call) 지원**
* **NETCONF Notification 지원**

📌 **YANG은 네트워크 장비 및 서비스를 정의하는 데이터 모델을 만들기 위해 설계된 언어임!**

**2️ 문서 상태 (Status of This Memo)**

🔹 **RFC 6020은 "Internet Standards Track" 문서**  
🔹 **IETF에서 공식 표준으로 채택**  
🔹 **IETF 커뮤니티의 합의를 통해 승인되었으며, 공개 검토를 거쳐 출판됨**

📌 **인터넷 표준 관련 세부 정보는 RFC 5741에서 확인 가능**

**3️ 저작권 및 라이선스 (Copyright Notice)**

🔹 **2010년 IETF Trust 및 저자가 저작권 보유**  
🔹 **IETF 문서 라이선스 규정을 따름 (BCP 78 및 IETF Trust Legal Provisions 적용)**  
🔹 **해당 문서의 코드 컴포넌트(Code Components)는 "Simplified BSD License" 하에 제공됨**

📌 **RFC 문서는 자유롭게 참조 가능하지만, 특정 변경 및 파생 작업에는 제한이 있을 수 있음**

**🔹 결론 (Summary)**

📌 **YANG은 NETCONF와 함께 사용되는 표준 데이터 모델링 언어**  
📌 **IETF 공식 표준이며, 네트워크 장비 및 서비스의 설정/상태를 모델링하는 데 필수적**  
📌 **라이선스 조항을 준수하여 자유롭게 사용 가능하지만, 일부 제한 사항 존재**

🚀 **NETCONF를 활용한 네트워크 자동화 및 관리에서는 YANG이 필수적인 요소!**

# 목차

[목차 2](#_Toc190265794)

[1. Introduction 4](#_Toc190265795)

[2. Keywords 5](#_Toc190265796)

[3. Terminology 6](#_Toc190265797)

[3.1 Mandatory Modes 7](#_Toc190265798)

[4. YANG Overview 8](#_Toc190265799)

[4.1 Functional Overview 8](#_Toc190265800)

[4.2 Language Overview 9](#_Toc190265801)

[4.2.1 Modules and Submodules 10](#_Toc190265802)

[4.2.2 Data Modeling Basics 11](#_Toc190265803)

[4.2.3 State Data 19](#_Toc190265804)

[4.2.4 Built-In Types 20](#_Toc190265805)

[4.2.5 Derived Types (typedef) 22](#_Toc190265806)

[4.2.6 Reusable Node Groups (grouping) 22](#_Toc190265807)

[4.2.7 Choices 24](#_Toc190265808)

[4.2.8 Extending Data Models (augment) 25](#_Toc190265809)

[4.2.9 RPC Definitions 27](#_Toc190265810)

[4.2.10 Notification Definitions 28](#_Toc190265811)

[5. Security Considerations 30](#_Toc190265812)

[6. IANA Considerations 33](#_Toc190265813)

[7. References 35](#_Toc190265814)

[9.1 Normative References 35](#_Toc190265815)

[9.2 Informative References 35](#_Toc190265816)

[Appendix A. Changes from RFC 4742 36](#_Toc190265817)

# 1. Introduction

**📌 YANG 소개 (Introduction)**

✅ **YANG = NETCONF을 위한 데이터 모델링 언어**  
✅ **구성(Configuration) 데이터, 상태(State) 데이터, RPC, 알림(Notification) 모델링 지원**  
✅ **XML 형식으로 데이터 모델 표현**

**1️ YANG이란?**

🔹 **YANG (Yet Another Next Generation) = NETCONF을 위한 데이터 모델링 언어**  
🔹 **NETCONF과 함께 사용되어 네트워크 장비 및 서비스의 데이터를 구조화하고 관리**  
🔹 **지원 기능:**

* **구성 데이터 (Configuration Data) 모델링**
* **상태 데이터 (State Data) 모델링**
* **NETCONF 원격 프로시저 호출 (Remote Procedure Call, RPC) 모델링**
* **NETCONF 알림 (Notification) 모델링**

📌 **YANG은 네트워크 장비 및 서비스의 설정, 모니터링, 운영 자동화를 지원하는 필수적인 요소!**

**2️ YANG과 NETCONF의 관계**

🔹 **YANG은 NETCONF 프로토콜의 "Operations Layer"와 "Content Layer"를 모델링하는 역할**  
🔹 **NETCONF 프로토콜과 함께 동작하며, 네트워크 장비의 설정 및 데이터를 관리하는 데 사용됨**

📌 **NETCONF 구성 프로토콜 (**[**RFC 4741**](#_13.1_Normative_References)**, Section 1.1)에서 NETCONF의 동작 방식과 관계 정의**

**3️ YANG 문서에서 다루는 내용**

📌 **이 문서(RFC 6020)는 다음을 설명:**  
🔹 **YANG의 문법(Syntax) 및 의미(Semantics)**  
🔹 **YANG 모듈에서 정의된 데이터 모델이 XML로 표현되는 방식**  
🔹 **NETCONF 작업을 통해 YANG 데이터 조작하는 방법**

**🔹 결론 (Summary)**

📌 **YANG은 NETCONF과 함께 사용되는 데이터 모델링 언어**  
📌 **네트워크 장비의 설정 및 상태 데이터를 표현하고, NETCONF을 통해 이를 조작할 수 있도록 설계됨**  
📌 **XML 기반 표현을 사용하며, RFC 6020에서는 YANG의 문법과 활용 방법을 설명함**

🚀 **네트워크 자동화 및 구성 관리를 위해 YANG + NETCONF는 필수적인 조합!**

# 2. Keywords

이 문서에서 사용되는 특정 키워드는 \*\*RFC 2119 (BCP 14)\*\*에 따라 해석되어야 합니다.

✅ **필수 요구사항 (강제적 규칙)**

* **MUST / REQUIRED / SHALL** → 반드시 준수해야 함
* **MUST NOT / SHALL NOT** → 절대 허용되지 않음

✅ **권장 사항 (비강제적 규칙)**

* **SHOULD / RECOMMENDED** → 강력히 권장되지만 특정 상황에서 예외 가능
* **SHOULD NOT / NOT RECOMMENDED** → 특별한 이유가 없는 한 사용하지 않아야 함

✅ **선택 사항 (자율적 규칙)**

* **MAY / OPTIONAL** → 선택적 사항, 필요에 따라 사용 가능

📌 **즉, 이 문서에서 위 키워드는 명확한 기술적 요구사항을 정의하는 데 사용됨!**

# 3. Terminology

**📌 YANG에서의 주요 용어들**

* **anyxml**: XML 데이터를 포함할 수 있는 데이터 노드로, 데이터의 구조를 모를 때 사용됩니다.
* **augment**: 기존의 스키마 노드에 새로운 노드를 추가하는 방법입니다.
* **base type**: 파생된 타입의 기본 타입으로, 내장 타입 또는 다른 파생 타입이 될 수 있습니다.
* **built-in type**: YANG에서 정의된 내장 데이터 타입 (예: uint32, string).
* **choice**: 여러 대체 중 하나만 유효한 스키마 노드입니다.
* **configuration data**: 시스템을 초기 상태에서 현재 상태로 변환하기 위해 필요한 데이터를 말합니다.
* **conformance**: 디바이스가 데이터 모델을 얼마나 정확히 따르는지를 나타내는 측정 기준입니다.
* **container**: 데이터 트리에서 하나의 인스턴스만 존재하는 내부 데이터 노드로, 값을 가지지 않지만 자식 노드들을 가질 수 있습니다.
* **data definition statement**: 데이터 노드를 정의하는 명령어로, 다양한 타입이 포함될 수 있습니다.
* **data model**: 데이터가 어떻게 표현되고 접근되는지를 설명하는 모델입니다.
* **data node**: 데이터 트리에서 인스턴스화할 수 있는 스키마 트리의 노드입니다.
* **data tree**: 디바이스에서 설정 및 상태 데이터를 표현하는 트리 구조입니다.
* **derived type**: 내장 타입 또는 다른 파생 타입에서 파생된 데이터 타입입니다.
* **device deviation**: 디바이스가 데이터 모델을 충실히 구현하지 않는 경우입니다.
* **extension**: YANG 명령어에 비-YANG 의미를 부여하는 확장 기능입니다.
* **feature**: 데이터 모델의 일부를 선택적으로 사용할 수 있게 하는 메커니즘입니다.
* **grouping**: 여러 스키마 노드를 재사용 가능하게 정의한 세트입니다.
* **identifier**: YANG 항목들을 이름으로 구별하는 방법입니다.
* **instance identifier**: 데이터 트리에서 특정 노드를 식별하는 방법입니다.
* **interior node**: 리프 노드를 제외한 트리 구조의 노드입니다.
* **leaf**: 하나의 인스턴스만 존재하는 데이터 노드로, 값을 가지며 자식 노드는 없습니다.
* **leaf-list**: 여러 인스턴스를 가지는 값들을 정의하는 데이터 노드입니다.
* **list**: 여러 인스턴스를 가질 수 있는 내부 데이터 노드입니다.
* **module**: YANG 모듈은 NETCONF 기반 작업을 위한 노드 계층을 정의합니다.
* **RPC**: NETCONF 프로토콜에서 원격 프로시저 호출을 의미합니다.
* **RPC operation**: NETCONF 프로토콜 내에서 특정한 원격 프로시저 호출을 의미합니다.
* **schema node**: 스키마 트리 내에서 하나의 노드를 나타냅니다.
* **schema tree**: 모듈 내에서 정의된 노드 계층입니다.
* **state data**: 읽기 전용 상태 정보나 수집된 통계 등의 설정 외 추가 데이터입니다.
* **submodule**: 다른 모듈에 파생된 타입, 그룹화, 데이터 노드 등을 추가하는 부분 모듈 정의입니다.
* **top-level data node**: 모듈이나 서브모듈 문서와 직결된 데이터 노드입니다.
* **uses**: 그룹화된 스키마 노드 세트를 인스턴스화하고 이를 특정 요구에 맞게 조정하는 명령어입니다.

💡 **YANG**의 각 용어들은 데이터 모델링과 관련된 핵심 개념들을 명확히 정의하여 네트워크 구성과 상태 데이터의 표현 및 조작을 지원합니다.

## 3.1 Mandatory Modes

**필수 노드**는 반드시 존재해야 하는 YANG 데이터 모델의 구성 요소를 의미합니다. 필수 노드는 다음 중 하나에 해당합니다:

1️ **Leaf, Choice, Anyxml 노드**

* "mandatory" 속성이 **true** 로 설정된 경우 필수 노드가 됩니다.

2️ **List 또는 Leaf-list 노드**

* "min-elements" 속성이 **0보다 큰 값**으로 설정된 경우 필수 노드가 됩니다.
* 즉, 최소한 하나 이상의 항목이 반드시 존재해야 합니다.

3️ **Container 노드 (Presence 속성 없음)**

* "presence" 문장이 정의되지 않은 경우,
* 그리고 적어도 하나 이상의 **필수 노드(mandatory node)**를 포함하고 있는 경우 필수 노드가 됩니다.

💡 **요약**

* 필수 노드는 NETCONF 클라이언트가 반드시 제공해야 하는 요소이며, 이를 만족하지 않으면 데이터 모델이 유효하지 않게 됩니다.
* "mandatory true" 또는 "min-elements > 0" 같은 조건을 통해 특정 노드가 필수임을 강제할 수 있습니다.
* "presence" 속성이 없는 container는 내부에 필수 노드가 존재하면 자동으로 필수 노드가 됩니다.

# 4. YANG Overview

## 4.1 Functional Overview

🔹 **YANG은 NETCONF 프로토콜을 위한 데이터 모델링 언어**

* YANG 모듈은 **구성 데이터(Configuration Data)**, **상태 데이터(State Data)**,  
  **원격 프로시저 호출(RPCs)**, **알림(Notifications)** 등을 포함하는 **데이터 계층 구조**를 정의함.
* NETCONF 클라이언트와 서버 간의 **데이터 송수신을 전체적으로 기술**할 수 있음.

🔹 **YANG 데이터 구조: 계층적 트리(Tree) 모델**

* 데이터는 **트리 구조**로 모델링되며, 각 노드는
  + 이름(Name)
  + 값(Value) **또는**
  + 하위 노드(Child Nodes)를 가짐.
* 노드 간의 관계와 동작을 명확하게 설명할 수 있음.

🔹 **모듈(Module)과 서브모듈(Submodule) 구조**

* **모듈은 자체적인 데이터 계층을 정의**하고,
  + 외부 모듈의 데이터를 **import**하여 사용 가능
  + **서브모듈(submodule)을 포함(include)하여** 데이터 계층 확장 가능
* 한 모듈에서 **다른 모듈의 데이터 계층을 증강(Augment)** 할 수도 있음.
  + 조건부 증강(Conditional Augmentation)도 가능 → 특정 조건이 충족될 때만 새로운 노드 추가.

🔹 **제약 조건 (Constraints) 정의 가능**

* 특정 노드가 **존재 여부**나 **값 범위**를 가지도록 제한할 수 있음.
* **클라이언트 또는 서버가 이러한 제약을 준수해야 함(MUST).**

🔹 **내장 데이터 유형 (Built-in Types) 및 파생 유형 (Derived Types)**

* 기본 제공되는 데이터 타입이 있으며, 이를 기반으로  
  **제약 조건(range, pattern 등)**을 설정하여 새로운 타입을 만들 수 있음.
* 예: uint32 타입에서 0~255 범위를 가지는 파생 타입을 정의 가능.

🔹 **노드 그룹화 (Groupings) 기능 제공**

* **재사용 가능한 노드 집합**을 정의하여 다른 모듈에서 사용 가능.
* 인스턴스화할 때 노드들을 **세부 조정(Refine) 또는 증강(Augment)** 할 수 있음.

🔹 **목록(List) 데이터 구조 지원**

* 리스트(List) 내부 항목은 **고유 키(Key) 값**으로 구분됨.
* 사용자가 직접 정렬(User-sorted)할 수도 있고, 시스템이 자동 정렬(System-sorted)할 수도 있음.
* **사용자 정렬 리스트의 경우 항목 순서를 변경하는 연산이 정의됨.**

🔹 **YIN (YANG Independent Notation) 변환 가능**

* **XML 기반의 YIN 표현식으로 변환 가능**, 즉 XML Parser와 XSLT에서 사용 가능.
* **양방향 변환(무손실 변환, Lossless Conversion)이 가능** → YIN에서 YANG으로 다시 변환 가능.

🔹 **확장성 (Extensibility) 지원**

* **표준 기구, 벤더, 개인이 새로운 확장 문법을 정의 가능**
* **기존 YANG 문법과 자연스럽게 공존하면서도 확장 문법을 눈에 띄게 표현 가능.**

🔹 **YANG은 NETCONF에 최적화된 데이터 모델링 언어**

* **임의의 XML 문서나 데이터 모델을 표현하는 것은 목표가 아님**
* NETCONF가 처리할 수 있는 형식으로 모델링되는 것을 **우선적으로 고려**

🔹 **SNMP SMIv2 (MIB 모듈)와의 호환성 유지**

* SNMP의 SMIv2(Structure of Management Information v2)와 **가능한 한 호환성을 유지**
* SMIv2 기반의 MIB 모듈을 YANG으로 변환 가능(단, **역변환은 지원하지 않음**).

🔹 **기존 장비 관리 시스템과 원활한 통합 가능**

* **장비의 네이티브 관리 인프라와 자연스럽게 연동**
* 기존의 **접근 제어 메커니즘을 활용하여 데이터 모델 보호 또는 공개 가능.**

✅ **요약: YANG은 NETCONF를 위한 강력한 데이터 모델링 언어!**  
✔ **트리 구조**로 데이터 계층 표현  
✔ **제약 조건, 데이터 타입, 그룹화, 확장성 지원**  
✔ **NETCONF 및 SNMP와 연동하여 장비 관리를 효율적으로 수행**

## 4.2 Language Overview

이 섹션에서는 **YANG의 핵심 개념**을 소개하여 이후 상세한 문법과 구문(Section 7) 이해를 돕습니다.  
YANG의 개념들은 상호 연관성이 강하기 때문에, **점진적인 접근 방식**을 사용하여 설명합니다.

**🔹 YANG의 주요 개념**

* YANG은 **데이터 모델링 언어**로서 **NETCONF** 프로토콜에서 사용됨.
* **모듈(Module)과 서브모듈(Submodule)**을 활용하여 데이터 구조를 정의.
* 데이터 계층을 **트리(Tree) 구조**로 모델링하며, 각 노드는 **이름(name)과 값(value)** 또는 **하위 노드(children)**를 가짐.
* **데이터 검증을 위한 제약 조건(Constraints)**을 정의할 수 있음.
* XML과의 호환성을 제공하며, **YANG Independent Notation (YIN)**으로 변환 가능.
* 확장성이 뛰어나며, **사용자 정의 확장(Extensions)**을 지원함.

### 4.2.1 Modules and Submodules

YANG에서 데이터 모델을 정의하는 기본 단위는 **모듈(Module)** 입니다. 모듈은 **NETCONF 서버에서 데이터를 구성하고 관리**하는 데 사용됩니다.

**🔹 모듈(Module)의 구성 요소**

모듈에는 **세 가지 유형의 문(statement)** 이 포함됩니다.

1. **모듈 헤더(Module-header)**
   * 모듈 자체에 대한 기본 정보를 제공
   * 예: 모듈 이름, 네임스페이스, 조직, 컨택트 정보
2. **리비전(Revision)**
   * 모듈의 **변경 이력(Version History)**을 설명
   * 각 리비전에는 **날짜, 설명, 변경 사항** 등이 포함됨
3. **정의(Definition)**
   * **데이터 모델을 실제로 정의**하는 부분
   * 컨테이너, 리스트, 리프(leaf) 등의 **데이터 구조**를 명시

**🔹 NETCONF 서버에서 모듈 사용 방식**

* NETCONF 서버는 **여러 개의 모듈을 구현**할 수 있음.
  + 같은 데이터를 **여러 개의 시각으로 제공** 가능
  + 또는 **각 모듈이 별도의 데이터 서브셋**을 담당할 수도 있음
* 단일 모듈로도 모든 데이터를 정의할 수 있음.

**🔹 서브모듈(Submodules)의 역할**

* **큰 모듈을 더 작은 단위**로 나누기 위해 사용됨.
* **외부에서 볼 때는 여전히 하나의 모듈**로 동작함.
* 모듈을 서브모듈로 나눌지는 **설계자의 선택**에 따라 결정됨.

**🔹 모듈 간 관계**

✅ **include**

* 모듈이 **자신의 서브모듈을 포함**할 때 사용
* 같은 모듈 내에서 **재사용성**을 높이는 데 유용

✅ **import**

* **외부 모듈의 데이터**를 참조할 때 사용
* 예를 들어, 한 모듈에서 **공통 데이터 모델을 불러올 때** import 사용 가능

**📝 요약**

* **모듈**: NETCONF에서 데이터를 정의하는 기본 단위
* **서브모듈**: 모듈을 더 작은 단위로 나눈 것 (외부에서는 여전히 하나의 모듈로 보임)
* **include**: 서브모듈을 포함하는 키워드
* **import**: 외부 모듈을 불러오는 키워드

✅ **이해 포인트**: **모듈을 어떻게 나누고, 참조하는지**가 NETCONF 기반 네트워크 관리에서 중요함. 🚀

### 4.2.2 Data Modeling Basics

YANG은 **NETCONF 프로토콜에서 사용할 데이터 모델을 정의**하는 언어입니다. YANG을 사용하여 **구성 데이터(configuration data), 상태 데이터(state data), RPCs, 알림(notifications)** 등을 모델링할 수 있습니다.

**🔹 YANG의 네 가지 주요 노드 유형 (Four Types of Nodes)**

YANG에서 데이터 모델링을 위해 사용되는 **네 가지 노드 유형**이 있습니다.

1️ **컨테이너 (Container)**  
2️ **리프 (Leaf)**  
3️ **리프 리스트 (Leaf-list)**  
4️ **리스트 (List)**

각 노드의 설명과 예제는 아래에서 다룹니다.

**🔹 1. 컨테이너(Container) 노드**

* 여러 개의 **자식 노드(child nodes)**를 그룹화하는 역할
* 자체적인 값(value)은 없음 (오직 **자식 노드**를 가짐)

**✅ YANG 예제**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✅ **NETCONF XML 표현**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 2. 리프(Leaf) 노드**

* **단일 값을 가지는 노드** (자식 노드 없음)
* 문자열(string), 정수(uint32), 부울(boolean) 등의 **데이터 타입을 지정** 가능

**✅ YANG 예제**

스크린샷, 텍스트, 블랙이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✅ **NETCONF XML 표현**



**🔹 3. 리프 리스트(Leaf-list) 노드**

* **여러 개의 값을 가질 수 있는 리프 노드**
* 각 값은 리스트처럼 저장됨 (단, 중복이 허용될 수도 있음)

**✅ YANG 예제**

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✅ **NETCONF XML 표현**



**🔹 4. 리스트(List) 노드**

* **반복되는 데이터 구조를 정의** (객체 배열 같은 개념)
* 각 항목(item)은 **키(key)**를 사용하여 구분됨

**✅ YANG 예제**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✅ **NETCONF XML 표현**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**📝 요약**

* **컨테이너(Container)**: **자식 노드 그룹화** (자체 값 없음)
* **리프(Leaf)**: 단일 값을 가지는 노드
* **리프 리스트(Leaf-list)**: **여러 개의 값을 저장**하는 리프
* **리스트(List)**: **반복적인 데이터 구조**를 정의하는 노드 (키 사용)

✅ **이해 포인트**: NETCONF에서 YANG 모델이 XML 형태로 변환되어 데이터를 표현하는 방식을 이해하는 것이 중요합니다! 🚀

#### 4.2.2.1 Leaf Nodes

**리프(Leaf) 노드**는 **단순한 데이터(예: 정수, 문자열 등)**를 저장하는 노드입니다.

* **자식 노드를 가질 수 없음**
* **단 하나의 값**만 저장 가능

🔹 YANG 리프(Leaf) 노드 예제

스크린샷, 텍스트, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ 위의 예제에서 host-name 노드는 **문자열(string) 타입의 단일 값**을 가집니다.

**🔹 NETCONF XML 표현**

YANG 모델이 NETCONF에서 XML 형식으로 변환되면 다음과 같이 표현됩니다.



☝ host-name 태그 안에 **단일 문자열 값** "my.example.com" 이 저장됩니다.

**📝 요약**

✅ **리프(Leaf) 노드 특징**  
✔ 단순 데이터 저장 (정수, 문자열 등)  
✔ 자식 노드 없음  
✔ **단 하나의 값**만 저장 가능

✅ **이해 포인트**  
YANG에서 정의된 leaf 노드는 **NETCONF에서 XML 태그로 변환**되어 전달됩니다. 🚀

#### 4.2.2.2 Leaf-List Nodes

**Leaf-List 노드**는 **하나의 데이터 타입을 가지는 다중 값 리스트**입니다.

* 여러 개의 **leaf 값을 포함**할 수 있음
* **각 값은 개별적인 leaf 노드처럼 취급됨**
* 동일한 데이터 타입을 유지해야 함

🔹 YANG Leaf-List 노드 예제

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ domain-search는 **여러 개의 문자열 값을 가질 수 있는 리스트**입니다.

**🔹 NETCONF XML 표현**

Leaf-List 노드가 **NETCONF XML 형식으로 변환**되면, **각 값이 개별 태그로 표현**됩니다.



☝ domain-search 태그가 **반복적으로 사용되며, 각 태그는 하나의 값을 가짐**.

**📝 요약**

✅ **Leaf-List 노드 특징**  
✔ **여러 개의 값**을 가질 수 있음  
✔ 각 값은 **개별적인 leaf 노드처럼 저장**됨  
✔ **동일한 데이터 타입을 유지**해야 함

✅ **이해 포인트**  
leaf-list는 **리스트 형태의 데이터**를 표현할 때 사용되며, **각 값은 개별 태그로 XML에서 변환**됩니다. 🚀

#### 4.2.2.3 Container Nodes

**Container 노드**는 **관련된 데이터를 그룹화하는 역할**을 합니다.

* **자식 노드를 포함하지만, 자체적인 값(value)은 없음**
* 자식 노드는 **leaf, list, container, leaf-list 등 모든 유형 가능**
* XML에서 **계층 구조를 형성**하는 데 유용

🔹 YANG Container 노드 예제

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ system 컨테이너는 **login 컨테이너를 포함**하며, login 컨테이너 내부에는 message라는 **leaf 노드가 존재**합니다.

**🔹 NETCONF XML 표현**

Container 노드는 **XML에서 계층적 구조로 변환**됩니다.

스크린샷, 텍스트, 폰트, 블랙이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ system 노드 내부에 login 노드가 있고, 그 내부에 message 노드가 포함된 **트리 구조**가 형성됩니다.

**📝 요약**

✅ **Container 노드 특징**  
✔ **그룹화된 데이터를 표현**하기 위해 사용됨  
✔ 자체적으로 **값(value)은 없고, 오직 자식 노드만 포함**  
✔ **자식 노드는 leaf, list, container 등 다양한 타입 가능**  
✔ **XML에서 계층적 구조를 표현하는 데 유용**

✅ **이해 포인트**  
container는 **데이터 구조를 논리적으로 그룹화하는 데 사용**되며, **XML에서도 같은 계층 구조로 표현**됩니다. 🚀

#### 4.2.2.4 List Nodes

**List 노드**는 **여러 개의 엔트리를 포함하는 시퀀스를 정의**합니다.

* **각 엔트리는 구조체(structure)나 레코드(record)와 유사**
* **key leafs 값이 고유한 식별자 역할**
* **여러 개의 key를 가질 수도 있음**
* 내부에 **leaf, list, container 등 다양한 노드를 포함 가능**

🔹 YANG List 노드 예제

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ user라는 리스트에는 name, full-name, class라는 **세 개의 leaf 노드**가 포함됨.  
☝ key "name"; → name 필드가 **각 사용자(user)의 고유 식별자 역할**을 함.

**🔹 NETCONF XML 표현**

List 노드는 **각 리스트 엔트리를 개별 XML 블록으로 표현**합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

☝ <user> 태그는 각 사용자를 나타내며, 내부에 name, full-name, class 정보를 포함.  
☝ name 필드가 **각 사용자를 식별하는 key 역할**을 수행.

**📝 요약**

✅ **List 노드 특징**  
✔ **각 항목이 key 값을 통해 고유하게 식별됨**  
✔ **리스트 내부에는 leaf, list, container 등 다양한 노드 포함 가능**  
✔ **XML에서는 여러 개의 <user> 태그로 변환**  
✔ **구조체(structure)나 테이블(row)처럼 사용 가능**

✅ **이해 포인트**  
list 노드는 **중복되지 않는 key 값으로 항목을 식별**하며, **XML에서는 각 항목이 개별 블록으로 표현**됩니다. 🚀

#### 4.2.2.5 Example Module

이 섹션에서는 YANG 모듈을 정의하는 예시를 제공합니다. 이 모듈은 **ACME 시스템**을 나타내며, 여러 **leaf**, **leaf-list**, **container**, **list** 노드를 결합하여 구성됩니다.

🔹 YANG 모듈 예시

텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 주요 요소들 설명**

* **module**: acme-system이라는 모듈 정의
  + **namespace**: 모듈의 네임스페이스를 http://acme.example.com/system로 정의
  + **prefix**: 이 모듈을 참조할 때 사용할 접두사 "acme"
  + **organization** 및 **contact**: 이 모듈을 만든 조직과 연락처 정보
  + **description**: 모듈에 대한 설명 제공
* **revision**: 모듈의 첫 번째 수정일인 2007-06-09를 정의

**🔹 데이터 구조**

* **container system**:
  + **leaf host-name**: 시스템의 호스트 이름을 나타내는 단일 값
  + **leaf-list domain-search**: 도메인 검색 목록 (여러 도메인 이름을 저장)
* **container login**:
  + **leaf message**: 로그인 세션 시작 시 표시되는 메시지
* **list user**:
  + 각 **user**는 name을 고유한 키로 하여 사용자 정보를 저장
  + **leaf name, full-name, class**: 각 사용자의 이름, 전체 이름, 클래스 정보를 담고 있음

**🔹 NETCONF XML 표현 예시**

이 YANG 모듈에 대한 NETCONF XML 표현은 다음과 같은 구조를 가질 수 있습니다:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**📝 요약**

✅ **모듈 정의**

* 모듈은 **module** 키워드로 시작하며, 다양한 **container**, **leaf**, **list** 등을 사용하여 데이터 구조를 정의합니다.

✅ **system** 및 **login** 컨테이너는 이 시스템의 주요 설정 및 로그인 관련 정보를 담고 있으며, **user** 리스트는 사용자별 정보를 정의합니다.

이 예시는 YANG 모듈의 기본적인 구성을 잘 보여줍니다. 다양한 노드를 결합하여 **복잡한 데이터 구조**를 정의하는 방법을 이해할 수 있습니다.

### 4.2.3 State Data

YANG은 상태 데이터를 구성 데이터와 함께 모델링할 수 있습니다. 이는 **config** 문을 기반으로 합니다. 노드가 **config false**로 표시되면 해당 서브 계층은 상태 데이터로 플래그가 지정됩니다. 이 상태 데이터는 **NETCONF**의 <get> 연산을 사용하여 보고되며, <get-config> 연산으로는 보고되지 않습니다. 상위 **container**, **list**, **key leaf**도 보고되며, 이는 상태 데이터의 맥락을 제공합니다.

**🔹 예시 설명**

이 예시에서는 각 인터페이스에 대해 두 개의 **leaf**를 정의합니다: 하나는 **configured speed**(구성된 속도)이고, 다른 하나는 **observed speed**(관찰된 속도)입니다. **observed speed**는 구성 데이터가 아니므로 \*\*config false\*\*로 표시됩니다. 이로 인해 \*\*observed speed\*\*는 **NETCONF**의 <get> 연산으로 반환될 수 있지만, <get-config> 연산에서는 반환되지 않습니다. 또한, \*\*observed speed\*\*는 **구성 데이터**가 아니므로 **<edit-config>** 연산을 통해 수정할 수 없습니다.

🔹 YANG 예시

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 주요 개념**

1. **config false**: 이 문은 노드를 상태 데이터로 설정합니다. 이를 통해 이 노드는 **<get>** 연산에서는 반환될 수 있지만, \*\*<get-config>\*\*에서는 반환되지 않습니다. 또한, 이 노드는 **편집되지 않음**(예: edit-config로 수정 불가).
2. **상태 데이터**: 상태 데이터는 장치의 현재 상태를 나타내는 정보입니다. 이 데이터는 사용자나 시스템에 의해 직접 수정되지 않으며, 일반적으로 읽기 전용입니다.

**🔹 NETCONF XML 예시**

이 YANG 모듈에 대한 **NETCONF XML 표현**은 다음과 같습니다:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

* **observed-speed** 값은 상태 데이터이므로, **<get>** 연산을 통해 읽을 수 있습니다.
* 하지만, **<get-config>** 연산을 통해 \*\*observed-speed\*\*는 반환되지 않습니다.

**📝 요약**

✅ \*\*config false\*\*를 사용하여 **상태 데이터**를 정의하고, 이를 **NETCONF**에서 어떻게 처리할지 명확히 구분합니다.  
✅ 상태 데이터는 **읽기 전용**이며, **구성 변경**을 통해 수정할 수 없습니다.

이러한 구분은 네트워크 장치에서 관리 및 상태 정보를 분리하여 효율적인 데이터 관리를 가능하게 합니다.

### 4.2.4 Built-In Types

YANG에는 많은 프로그래밍 언어와 유사한 **내장 데이터 타입**이 존재하지만, 관리 도메인에서의 특별한 요구 사항에 맞게 일부 차이가 있습니다. 다음 표는 **내장 타입**을 요약한 것입니다:

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 설명**

* **binary**: 이진 데이터를 나타내며, 어떤 형식의 데이터든지 이진 값으로 표현할 수 있습니다.
* **bits**: 비트 집합으로, 주로 플래그나 상태 정보에 사용됩니다.
* **boolean**: 참/거짓 값("true"/"false")을 나타냅니다.
* **decimal64**: 64비트 크기의 부호 있는 십진수로, 수학적 계산을 필요로 하는 경우 사용됩니다.
* **empty**: 값이 없는 leaf로, 이를 통해 값이 없음을 명시할 수 있습니다.
* **enumeration**: 미리 정의된 값들 중에서 선택되는 열거형 값을 사용합니다.
* **identityref**: 추상적 개념을 참조하는 타입으로, 실제 값은 다른 정의된 정체성으로부터 올 수 있습니다.
* **instance-identifier**: 특정 데이터 트리 노드를 가리키는 참조입니다.
* **intX, uintX**: 부호 있는 및 부호 없는 정수를 나타내며, 8비트에서 64비트까지 다양합니다.
* **leafref**: 다른 leaf 인스턴스를 참조하는 타입입니다.
* **string**: 일반적인 문자열 데이터로 사람이 읽을 수 있는 값을 나타냅니다.
* **union**: 여러 가능한 타입들 중 하나를 선택하는 데이터 타입입니다.

**🔹 관련 문서**

* **"type" statement**는 **Section 7.4**에서 다루어집니다.

이 **내장 타입들**은 YANG 데이터 모델링에서 다양한 종류의 데이터를 효율적으로 표현하는 데 도움을 주며, 관리와 구성이 가능한 방식으로 데이터의 구조와 의미를 명확하게 정의합니다.

### 4.2.5 Derived Types (typedef)

YANG은 **"typedef"** 문을 사용하여 **기본 타입**으로부터 **파생된 타입**을 정의할 수 있습니다. 기본 타입은 내장 타입이거나 파생된 타입일 수 있으며, 이를 통해 타입의 계층 구조를 형성할 수 있습니다.

파생된 타입은 **"type"** 문에 인수로 사용할 수 있습니다.

YANG 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

NETCONF XML 예시:



**🔹 설명**

* **typedef**: 새 타입을 정의하는 데 사용됩니다. 여기서 percent는 0에서 100 사이의 uint8 값을 갖는 파생 타입으로 정의되었습니다.
* **leaf**: completed라는 leaf는 percent 타입을 가지며, 이 값은 0과 100 사이의 부호 없는 8비트 정수로 정의됩니다.

**🔹 관련 문서**

* **"typedef" statement**는 **Section 7.3**에서 다루어집니다.

**파생 타입**을 사용하면 특정 범위나 규칙을 가진 새로운 타입을 정의할 수 있어 데이터 모델링의 유연성 및 재사용성을 높일 수 있습니다.

### 4.2.6 Reusable Node Groups (grouping)

YANG에서 **"grouping"** 문을 사용하여 여러 노드를 재사용 가능한 컬렉션으로 묶을 수 있습니다. **grouping**은 노드 세트를 정의하고, **"uses"** 문을 통해 해당 그룹을 인스턴스화하여 사용합니다.

YANG 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

NETCONF XML 예시:

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**설명**

* **grouping**: target라는 그룹을 정의하여 address와 port라는 두 개의 leaf 노드를 포함합니다.
* **uses**: peer 컨테이너 내에서 destination 컨테이너에 target 그룹을 사용합니다. 이 그룹은 정의된 address와 port 값을 포함하는 데이터 구조를 인스턴스화합니다.

이 예시에서, 동일한 그룹을 여러 컨테이너에서 재사용할 수 있게 됩니다.

**그룹 정제**

* 그룹은 \*\*"uses"\*\*로 사용될 때 세부 사항을 정제할 수 있습니다. 예를 들어, address와 port의 **description**을 수정하는 방법을 보여줍니다:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이렇게 하면 source와 destination에 대해 각각 다른 설명을 제공할 수 있습니다.

**🔹 관련 문서**

* **"grouping" statement**는 **Section 7.11**에서 다루어집니다.

**장점**

* **재사용성**: 동일한 그룹을 여러 위치에서 사용함으로써 코드의 중복을 줄이고 유지 관리가 용이해집니다.
* **유연성**: 그룹을 사용할 때 정제하여 구체적인 요구 사항에 맞게 수정할 수 있습니다.

### 4.2.7 Choices

YANG에서는 \*\*"choice"\*\*와 **"case"** 문을 사용하여 상호 배타적인 노드를 구분할 수 있습니다. choice 문은 여러 개의 case 문을 포함하고, 각 case는 함께 나타날 수 없는 스키마 노드 집합을 정의합니다. 각 case는 여러 노드를 포함할 수 있지만, 각 노드는 단 하나의 case에만 나타날 수 있습니다.

YANG 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

NETCONF XML 예시:



**설명**

* **choice**: snack라는 선택을 정의합니다. 이 선택에는 두 가지 case가 있습니다:
  + sports-arena: pretzel과 beer를 선택할 수 있습니다.
  + late-night: chocolate의 종류를 dark, milk, first-available로 선택할 수 있습니다.
* **case**: case는 choice 내에서 상호 배타적인 옵션을 정의합니다. 즉, sports-arena와 late-night는 동시에 나타날 수 없습니다. 하나의 case에서 요소가 생성되면, 다른 case에서의 모든 요소는 자동으로 삭제됩니다.

이와 같은 방식으로, 장치가 선택된 옵션에 대해 불일치가 발생하지 않도록 처리합니다.

**🔹 관련 문서**

* **"choice" statement**는 **Section 7.9**에서 다루어집니다.

**장점**

* **상호 배타적인 선택**: choice와 case를 통해 사용자가 특정 데이터를 선택할 때 다른 선택이 자동으로 제외되므로, 잘못된 데이터 조합을 방지할 수 있습니다.
* **개념적 단순화**: 추가적인 계층 구조 없이 데이터 모델의 개념적 구조를 유지하면서도 상호 배타적인 옵션을 관리할 수 있습니다.

### 4.2.8 Extending Data Models (augment)

YANG에서는 **"augment"** 문을 사용하여 데이터 모델에 추가적인 노드를 삽입할 수 있습니다. 이 문은 현재 모듈 및 그 하위 모듈 또는 외부 모듈에 노드를 삽입하는 데 유용합니다. 예를 들어, 벤더는 표준 데이터 모델에 벤더-특정 파라미터를 추가할 수 있습니다.

augment 문은 데이터 모델 계층에서 새 노드를 삽입할 위치를 정의하고, when 문은 새로운 노드가 유효할 조건을 정의합니다.

YANG 예시:

텍스트, 폰트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이 예시는 "class"가 "wheel"이 아닌 경우에만 유효한 "uid" 노드를 정의합니다.

NETCONF XML 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이 경우, 만약 이 확장이 "other"라는 접두어를 가진 모듈에 속한다면 XML에서 "uid" 요소는 other:uid로 표현됩니다.

**설명**

* **augment**: 기존의 데이터 모델에 노드를 삽입하는 기능을 제공합니다. 이 방법을 사용하면 외부 모듈이나 하위 모듈에 정의된 노드를 확장할 수 있습니다.
* **when**: 새로 삽입된 노드가 유효한 조건을 지정합니다. 이 예시에서는 class 값이 "wheel"이 아닐 때만 "uid" 노드가 유효합니다.

**🔹 관련 문서**

* **"augment" statement**는 **Section 7.15**에서 다루어집니다.

**장점**

* **모듈 확장**: 기존 데이터 모델을 수정하지 않고도 필요한 필드나 파라미터를 추가할 수 있습니다.
* **유연한 조건 정의**: when 문을 사용하여 특정 조건에서만 노드가 유효하도록 설정할 수 있어, 데이터 모델의 유효성을 세밀하게 관리할 수 있습니다.
* **벤더 확장**: 벤더는 표준 모델을 변경하지 않고 추가적인 벤더 고유의 파라미터를 삽입할 수 있어, 표준화된 인터페이스와 호환성을 유지하면서도 추가적인 기능을 제공할 수 있습니다.

### 4.2.9 RPC Definitions

YANG은 **NETCONF RPC**(Remote Procedure Call)를 정의할 수 있는 기능을 제공합니다. RPC의 이름, 입력 매개변수, 출력 매개변수는 YANG 데이터 정의 문을 사용하여 모델링됩니다. 이를 통해 시스템 간에 원격 절차 호출을 정의하고 관리할 수 있습니다.

YANG 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

이 예시에서 activate-software-image라는 RPC는 image-name이라는 입력을 받고, status라는 출력을 반환합니다.

NETCONF XML 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

위 XML 예시는 activate-software-image RPC 요청을 보내고, 그에 대한 응답으로 status 값을 받는 모습을 보여줍니다. 메시지 ID 101을 사용하여 요청과 응답을 매핑합니다.

**설명**

* **rpc**: YANG에서 rpc 문을 사용하여 원격 프로시저 호출을 정의합니다. 이 정의는 입력과 출력 매개변수뿐만 아니라 호출될 함수의 이름을 포함합니다.
* **input**: RPC가 처리할 입력 파라미터를 정의합니다. 이 예시에서는 image-name이라는 문자열 타입의 입력을 정의하고 있습니다.
* **output**: RPC의 결과로 반환될 출력 파라미터를 정의합니다. 이 예시에서는 status라는 문자열을 출력으로 정의합니다.
* **rpc-reply**: 요청에 대한 응답을 나타내며, 출력 매개변수를 포함합니다.

**🔹 관련 문서**

* **"rpc" statement**는 **Section 7.13**에서 다루어집니다.

**장점**

* **원격 프로시저 호출**: YANG을 사용하여 다양한 시스템 간에 RPC를 정의하고, 데이터 모델 내에서 명확히 지정된 방식으로 호출할 수 있습니다.
* **유연한 파라미터 정의**: 입력과 출력을 각각 정의하여 필요한 데이터를 정확히 주고받을 수 있으며, 이를 통해 시스템 간의 상호작용을 명확히 설정할 수 있습니다.
* **구체적인 관리**: NETCONF XML을 통해 RPC를 요청하고 응답을 받을 수 있으며, 메시지 ID를 사용해 요청과 응답을 추적할 수 있어 관리가 용이합니다.

### 4.2.10 Notification Definitions

YANG은 \*\*NETCONF 알림(Notification)\*\*을 정의할 수 있도록 지원합니다. 알림은 특정 이벤트가 발생했음을 클라이언트에게 전달하는 데 사용되며, YANG의 데이터 정의 문을 활용하여 알림의 내용을 모델링할 수 있습니다.

YANG 예시:

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**설명:**

* notification 문을 사용하여 **link-failure**라는 알림을 정의합니다.
* description 필드는 해당 알림이 무엇을 의미하는지 설명합니다.
* leaf if-name: 장애가 발생한 인터페이스의 이름을 leafref를 사용하여 참조합니다.
* leaf if-admin-status: 관리 상태(admin-status).
* leaf if-oper-status: 운영 상태(oper-status).

NETCONF XML 예시:

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**설명:**

* <notification> 요소: NETCONF 프로토콜에서 알림을 나타냅니다.
* <eventTime>: 알림이 발생한 시점을 ISO 8601 형식으로 기록합니다.
* <link-failure>: link-failure 이벤트가 발생했음을 나타냅니다.
* <if-name>: 장애가 발생한 인터페이스(so-1/2/3.0).
* <if-admin-status>: 인터페이스의 관리 상태 (up).
* <if-oper-status>: 운영 상태 (down).

**🔹 관련 문서**

* "notification" 문은 **Section 7.14**에서 다룹니다.

**🔹 장점**

* **실시간 이벤트 감지**: 장비에서 특정 이벤트(예: 링크 장애)가 발생했을 때, 이를 클라이언트에게 즉시 알릴 수 있습니다.
* **구조화된 데이터 모델**: YANG을 활용하면 알림 데이터를 체계적으로 정의할 수 있어, 다양한 네트워크 장비 간 호환성이 보장됩니다.
* **NETCONF 통합**: NETCONF <notification> 메시지를 통해 네트워크 관리 시스템(NMS)과 원활한 연동이 가능합니다.

YANG의 알림 기능은 네트워크 모니터링 및 장애 대응에 필수적인 역할을 하며, 실시간 운영 정보를 제공하여 빠른 문제 해결을 가능하게 합니다. 🚀

# 5. Language Concepts

YANG 언어의 모듈 및 서브모듈 개념은 데이터 모델을 정의하고 확장하는 핵심 요소입니다. 아래는 **모듈 및 서브모듈** 개념을 요약한 내용입니다.

## 5.1 Modules and Submodules

**📌 모듈 (Module)**

* YANG에서 데이터 모델을 정의하는 **기본 단위**.
* 독립적인 모델을 정의하거나, 기존 모델을 **확장(augment)** 가능.
* 다른 모듈을 **import** 하여 외부 정의를 참조 가능.
* 하나의 모듈은 **여러 서브모듈(submodules)을 포함**할 수 있음.

**📌 서브모듈 (Submodule)**

* 하나의 모듈을 구성하는 **부분 모듈**로, 단독 실행 불가능.
* 하나의 서브모듈은 **오직 하나의 모듈에만 속할 수 있음**.
* 모듈에서 include 문을 사용하여 서브모듈을 포함.

**📌 모듈 및 서브모듈의 네이밍 규칙**

* **표준 모듈 및 서브모듈의 이름은 반드시 유일해야 함**.
* 기업(enterprise) 모듈 개발자는 **충돌을 피하기 위해 조직명을 접두사(prefix)로 사용**할 것을 권장.
  + 예: cisco-ospf, juniper-bgp

### 5.1.1 Import and Include by Revision

* YANG 모듈은 시간이 지나면서 개정(revision)될 수 있으며, 모듈을 사용할 때 **특정 revision을 지정**해야 함.
* import는 다른 모듈을 참조할 때 사용.
* include는 **모듈이 자신의 서브모듈을 포함할 때 사용**.

**📌 예제: 모듈 Import**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

* b 모듈은 a 모듈을 import하고 prefix p로 참조.
* 특정 revision-date를 명시하여 해당 버전의 모듈을 사용.

### 5.1.2 Module Hierarchies

* YANG은 \*\*다중 계층 구조(Multi-hierarchy)\*\*를 지원.
* 하나의 YANG 모듈은 **여러 개의 최상위 노드**를 가질 수 있음.
* NETCONF에서 각 모듈의 최상위 노드는 <config> 또는 <data>의 **하위 요소**로 인코딩됨.

**📌 예제: 다중 최상위 노드를 가진 모듈**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

📌 NETCONF XML 예제

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

* <config> 내부에 여러 개의 \*\*최상위 컨테이너(system, routing)\*\*가 포함됨.
* 이는 YANG이 **유연한 데이터 모델링**을 지원함을 보여줌.

**🔹 핵심 정리**

✔ **모듈(Module)**: 독립적인 YANG 데이터 모델을 정의.  
✔ **서브모듈(Submodule)**: 하나의 모듈을 구성하는 부분으로, 단독 실행 불가.  
✔ **import**: 다른 모듈을 참조할 때 사용.  
✔ **include**: 서브모듈을 포함할 때 사용.  
✔ **revision 사용**: 특정 버전의 모듈을 사용하도록 명시 가능.  
✔ **다중 계층 구조 지원**: 하나의 모듈에서 여러 개의 최상위 노드를 가질 수 있음.

YANG의 모듈 및 서브모듈 개념을 잘 이해하면, 보다 **구조적이고 확장 가능한 네트워크 데이터 모델**을 설계할 수 있습니다. 🚀

## 5.2 File Layout

YANG 모듈과 서브모듈은 보통 **각각 별도의 파일**에 저장됩니다.  
파일명은 다음과 같은 형식을 따르는 것이 좋습니다.



🔹 파일명 예시

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 파일 레이아웃의 목적**

1. **모듈과 서브모듈을 쉽게 찾을 수 있도록 표준화**
   * import된 모듈이나 include된 서브모듈을 파일명 규칙을 통해 쉽게 찾을 수 있음.
2. **YANG 컴파일러의 효율성 향상**
   * YANG 컴파일러는 모듈을 독립적으로 처리하지만, 서브모듈은 **전체 모듈이 링크될 때 최종 검증됨**.
3. **버전 관리를 쉽게 하기 위해 개정(revision) 포함 가능**
   * 예를 들어, ospf@2023-06-01.yang과 ospf@2024-02-12.yang을 별도 관리 가능.

**🔹 실전 예제**

**📌 YANG 모듈 (routing.yang)**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

📌 YANG 서브모듈 (routing-submodule.yang)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**🔹 핵심 정리**

✔ **모듈과 서브모듈은 각각 별도의 파일로 저장**  
✔ **파일명은 "모듈이름[@개정날짜].yang/yin" 형태를 따르는 것이 좋음**  
✔ **컴파일러는 서브모듈을 독립적으로 처리할 수 있지만, 전체 모듈로 연결될 때 최종 검증됨**  
✔ **개정(revision) 날짜를 포함하면 버전 관리를 용이하게 할 수 있음**

파일 레이아웃을 표준화하면 YANG 모델을 **더 효율적이고 체계적으로 관리**할 수 있습니다. 🚀

## 5.3 XML Namespaces

YANG에서 정의된 모든 요소는 특정 \*\*XML 네임스페이스(namespace)\*\*에 바인딩됩니다.  
이는 **전역적으로 유일한 URI**를 사용하여 정의됩니다.  
👉 **NETCONF 클라이언트 및 서버는 XML 데이터 인코딩 시 해당 네임스페이스를 사용합니다.**

**🔹 XML 네임스페이스 개념**

* 네임스페이스(namespace)는 XML 요소의 충돌을 방지하는 \*\*고유 식별자(URI)\*\*입니다.
* YANG 모듈은 XML 네임스페이스를 사용하여 **서로 다른 데이터 모델을 구분**합니다.

**📌 예제: 네임스페이스가 있는 YANG 모듈**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

위의 namespace "http://example.com/network"; 선언은 **모듈 내의 모든 요소가 해당 네임스페이스에 속함**을 의미합니다.

📌 NETCONF XML 인코딩 예제

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✔ xmlns="http://example.com/network" 네임스페이스를 사용하여 device 요소가 특정 모듈에서 왔음을 명확히 합니다.

🔹 네임스페이스 할당 규칙

텍스트, 스크린샷, 폰트, 화이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

👉 **RFC를 통해 공식 등록된 모듈은 반드시 IANA에서 네임스페이스를 할당받아야 합니다.**  
👉 **기업 또는 개인이 정의한 모듈은 네임스페이스 충돌을 방지하기 위해 기업명이나 도메인을 포함하는 것이 권장됩니다.**

### 5.3.1 YANG XML Namespace

YANG은 \*\*NETCONF의 <edit-config> 및 <error-info>\*\*와 같은 **특정 XML 구조를 정의하는 네임스페이스**를 제공합니다.

**📌 YANG에서 사용하는 기본 XML 네임스페이스:**



✔ NETCONF 관련 XML 요소를 처리할 때 사용됩니다.

**📌 예제: YANG 기본 네임스페이스 적용**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

👉 xmlns="urn:ietf:params:xml:ns:yang:1"를 선언하여 **NETCONF의 YANG 관련 데이터 모델을 처리할 수 있도록 함**.

**🔹 핵심 정리**

✔ **YANG 모듈은 전역적으로 유일한 XML 네임스페이스(URI)를 사용하여 정의됨**  
✔ **RFC를 통해 공식 발표된 모듈은 IANA에서 네임스페이스를 할당받아야 함**  
✔ **기업이나 개인 모듈은 도메인명을 포함하여 네임스페이스 충돌을 방지해야 함**  
✔ **NETCONF 관련 YANG XML 네임스페이스: urn:ietf:params:xml:ns:yang:1**

이렇게 네임스페이스를 관리하면 **여러 모듈 간 충돌을 방지하고, YANG 모델의 확장성을 보장**할 수 있습니다. 🚀

## 5.4 Resolving Grouping, Type, and Identity Names

YANG에서는 **grouping, type, identity**와 같은 요소의 이름을 **정의된 컨텍스트에서 해결**합니다.  
즉, **이름이 사용되는 위치가 아니라, 정의된 위치에서 의미를 해석**합니다.  
👉 **이 방식은 일반적인 프로그래밍 언어에서의 정적 스코핑(static scoping)과 유사합니다.**

**🔹 1. 그룹핑(Grouping)과 타입(Type) 해석**

* **grouping**: 데이터 모델에서 재사용 가능한 구조를 정의합니다.
* **typedef**: 새로운 데이터 타입을 정의합니다.
* **이름 해석 시, 원래 정의된 컨텍스트에서 타입을 찾습니다.**

**📌 예제: 그룹핑과 타입의 컨텍스트 해석**

**📍 모듈 A: grouping 및 typedef 정의**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

📍 모듈 B: 모듈 A의 grouping 사용

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✔ **my-type은 b 모듈에서 따로 정의할 필요 없음**  
✔ **my-type은 원래 정의된 a 모듈의 컨텍스트에서 해석됨**

👉 **결과적으로, 그룹핑을 사용하는 모듈(b)에서 typedef를 재정의할 필요 없이 원본 모듈(a)에서 해결됩니다.**

**🔹 2. 아이덴티티(Identity) 해석**

* **identity**는 YANG의 열거형(enum) 개념과 유사하며, 확장이 가능합니다.
* **identity는 정의된 모듈의 컨텍스트에서 해석됩니다.**

**📌 예제: identity 해석**

**📍 모듈 A에서 identity 정의**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

📍 모듈 B에서 모듈 A의 identity 사용

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

✔ **모듈 b는 a:base-id를 import했기 때문에, identityref 타입에서 이를 사용 가능**  
✔ **해당 identity가 b 모듈에서 재정의될 필요 없음**

👉 **이 방식으로 모듈 간 identity를 확장할 수 있으며, 이름 충돌을 방지할 수 있음**

**🔹 3. 핵심 정리**

✔ **타입(type), 그룹핑(grouping), 아이덴티티(identity)는 "정의된" 컨텍스트에서 해결됨**  
✔ **사용하는 모듈이 아니라, "원래 정의된" 모듈에서 의미를 해석**  
✔ **모듈 간 참조(import/include) 시, 타입 재정의가 필요 없음**  
✔ **이 방식은 프로그래밍 언어의 정적 스코핑(static scoping)과 유사함**

👉 **이 원칙을 따르면 모듈 간 충돌을 방지하고, 데이터 모델의 일관성을 유지할 수 있습니다.** 🚀

## 5.5 Nested Typedefs and Groupings

# 6. IANA Considerations

✅ **NETCONF 및 RESTCONF Call Home을 위한 기본 포트가 IANA에 등록됨.**  
✅ **IANA(User Ports Range)에서 할당된 공식 포트 번호:**

📌 IANA 등록된 포트 정보

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**📌 포트 등록 정보 (IANA 등록 템플릿)**

**1️ NETCONF Call Home (SSH)**

* **서비스 이름:** netconf-ch-ssh
* **포트 번호:** 4334
* **전송 프로토콜:** TCP
* **설명:** NETCONF Call Home (SSH)
* **담당자:** IESG <iesg@ietf.org>
* **연락처:** IETF Chair <chair@ietf.org>
* **참조 문서:** RFC 8071

**2️ NETCONF Call Home (TLS)**

* **서비스 이름:** netconf-ch-tls
* **포트 번호:** 4335
* **전송 프로토콜:** TCP
* **설명:** NETCONF Call Home (TLS)
* **담당자:** IESG <iesg@ietf.org>
* **연락처:** IETF Chair <chair@ietf.org>
* **참조 문서:** RFC 8071

**3️ RESTCONF Call Home (TLS)**

* **서비스 이름:** restconf-ch-tls
* **포트 번호:** 4336
* **전송 프로토콜:** TCP
* **설명:** RESTCONF Call Home (TLS)
* **담당자:** IESG <iesg@ietf.org>
* **연락처:** IETF Chair <chair@ietf.org>
* **참조 문서:** RFC 8071

**📌 결론 (Summary)**

🔹 **IANA가 공식적으로 Call Home 기능을 위한 포트를 지정함.**  
🔹 **기본적으로 IANA 할당 포트(4334, 4335, 4336)를 사용해야 하지만, 필요하면 다른 포트로 변경 가능.**  
🔹 **IANA 등록 사항은 RFC 8071을 기반으로 관리됨.**

📌 **NETCONF 및 RESTCONF Call Home을 설정할 때, 해당 포트를 방화벽 및 네트워크 정책에서 허용해야 함!** 🚀

# 7. References

## 9.1 Normative References

[RFC 2119](https://datatracker.ietf.org/doc/html/rfc2119) Key words for use in RFCs to Indicate Requirement Levels

[RFC 3023](https://datatracker.ietf.org/doc/html/rfc3023) XML Media Types

[RFC 3629](https://datatracker.ietf.org/doc/html/rfc3629) UTF-8, a transformation format of ISO 10646

[RFC 3688](https://datatracker.ietf.org/doc/html/rfc3688) The IETF XML Registry

[RFC 3986](https://datatracker.ietf.org/doc/html/rfc3986) Uniform Resource Identifier (URI): Generic Syntax

[RFC 4648](https://datatracker.ietf.org/doc/html/rfc4648) The Base16, Base32, and Base64 Data Encodings

[RFC 4741](https://datatracker.ietf.org/doc/html/rfc4741) NETCONF Configuration Protocol

[RFC 5226](https://datatracker.ietf.org/doc/html/rfc5226) Guidelines for Writing an IANA Considerations Section in RFCs

[RFC 5234](https://datatracker.ietf.org/doc/html/rfc5234) Augmented BNF for Syntax Specifications: ABNF

[RFC 5277](https://datatracker.ietf.org/doc/html/rfc5277) NETCONF Event Notifications

[XML-NAMES](https://www.w3.org/TR/2009/REC-xml-names-20091208/) Namespaces in XML 1.0 (Third Edition)

[XPATH](https://www.w3.org/TR/1999/REC-xpath-19991116/) XML Path Language (XPath) Version 1.0

[XSD-TYPES](https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/) XML Schema Part 2: Datatypes Second Edition

## 9.2 Informative References

[RFC 2578](https://datatracker.ietf.org/doc/html/rfc2578) Structure of Management Information Version 2 (SMIv2)

[RFC 2579](https://datatracker.ietf.org/doc/html/rfc2579) Textual Conventions for SMIv2

[RFC 3780](https://datatracker.ietf.org/doc/html/rfc3780) SMIng - Next Generation Structure of Management Information

[RFC 4844](https://datatracker.ietf.org/doc/html/rfc4844) The RFC Series and RFC Editor

[XPATH2.0](https://www.w3.org/TR/xpath20/) XML Path Language (XPath) 2.0 (Second Edition)

[XSLT](https://www.w3.org/TR/1999/REC-xslt-19991116) XSL Transformations (XSLT) Version 1.0

# Appendix A. Changes from RFC 4742

이 문서는 **RFC 4742**와 비교하여 다음과 같은 주요 변경 사항을 포함함:

📌 **보안 강화**

* 새로운 **청크 프레이밍 메커니즘**(chunked framing mechanism) 도입하여 **EOM 프레이밍의 보안 문제 해결**.
* \*\*보안 고려 사항(Security Considerations)\*\*을 확장하고, **EOM 관련 문제**에 대한 설명 추가.

📌 **기술적 개선**

* \*\*청크 인코딩(chunked encoding)\*\*의 예시 추가 및 줄바꿈 위치 강조.
* **NETCONF 사용자명(username) 처리 관련 요구 사항**을 **RFC 6241**에 맞춰 수정.
* **"client/server" 및 "manager/agent"** 용어를 각각 \*\*"SSH client/server" 및 "NETCONF client/server"\*\*로 일관되게 변경.
* **"command" 또는 "message" 대신 "operation"** 용어를 일관되게 사용.

📌 **기타 정리**

* **RFC 4742에서 보고된 오류(errata) 반영**. (공식 오류 목록은 [RFC 에디터 웹사이트](http://www.rfc-editor.org)에서 확인 가능)