**Network Configuration Protocol (NETCONF)**

**문서 이름**: **Request for Comments (RFC) 6241**

**발행일**: 2011년 6월

**ISSN**: 2070-1721

**카테고리**: **Standards Track**

**관련 RFC**: 이 문서는 이전의 **RFC 4741**을 대체(Obsolete)합니다.

**NETCONF 프로토콜의 목적 및 주요 특징**

1. **목적**:
   * **NETCONF**는 네트워크 장치의 \*\*구성(configuration)\*\*을 설치, 조작, 삭제하기 위한 메커니즘을 제공합니다.
2. **데이터 형식**:
   * 구성 데이터 및 프로토콜 메시지를 **XML(Extensible Markup Language)** 기반 데이터 인코딩을 사용하여 표현합니다.
3. **운영 방식**:
   * NETCONF 프로토콜은 \*\*원격 프로시저 호출(Remote Procedure Calls, RPC)\*\*로 실현됩니다. 이를 통해 네트워크 장치 간의 통신 및 제어가 이루어집니다.

**문서의 상태와 역할**

* 이 문서는 **Internet Standards Track** 문서로서, 표준화된 기술을 정의합니다.
* \*\*IETF(Internet Engineering Task Force)\*\*의 산출물이며, IETF 커뮤니티의 합의에 기반합니다.
* \*\*IESG(Internet Engineering Steering Group)\*\*의 검토 및 승인을 거쳐 공개되었습니다.

**추가 정보**

* 문서의 상태, 수정 사항(Errata), 그리고 피드백 제공 방법은 다음의 URL에서 확인할 수 있습니다:  
  <http://www.rfc-editor.org/info/rfc6241>.

**NETCONF가 대체한 RFC**

* 이 문서는 이전 표준인 **RFC 4741**을 대체(Obsoletes)하여 NETCONF의 새로운 버전을 정의합니다. 이는 기존 프로토콜의 개선 및 최신 요구 사항을 반영하기 위함입니다.

**요약**

RFC 6241은 네트워크 구성 관리 프로토콜인 **NETCONF**에 대한 표준 정의 문서입니다. 이 프로토콜은 XML 기반 데이터를 사용하며, RPC 메커니즘을 통해 네트워크 장치를 효율적으로 제어하고 관리할 수 있도록 설계되었습니다. 이는 IETF의 공식 표준 문서로, 네트워크 구성 및 관리의 핵심 기술 중 하나로 자리 잡고 있습니다.

이 문서는 IETF Trust 및 문서 작성자의 저작권 아래 보호됩니다.  
문서를 사용하는 사람은 BCP 78 및 관련 법적 조항을 준수해야 하며, 수정 및 파생 작업은 **IETF Standards Process** 내에서만 허용됩니다. 특히, 문서에서 제공된 코드는 Simplified BSD License 하에 제공되며, 보증이 없습니다.

# 목차

[목차 3](#_Toc189582266)

[1. Introduction 3](#_Toc189582267)

[1.1 Terminology 4](#_Toc189582268)

[1.2 Protocol Overview 7](#_Toc189582269)

[1.3 Capabilities 9](#_Toc189582270)

[1.4 Separation of Configuration and State Data 10](#_Toc189582271)

[2. Transport Protocol Requirements 11](#_Toc189582272)

# 1. Introduction

NETCONF 프로토콜은 네트워크 장치를 관리하고, 구성 데이터를 검색하며, 새로운 구성 데이터를 업로드하고 조작할 수 있도록 하는 간단한 메커니즘을 정의합니다. 이 프로토콜을 사용하면 장치가 완전한 형식의 애플리케이션 프로그래밍 인터페이스(API)를 노출할 수 있으며, 이를 통해 애플리케이션이 전체 또는 부분적인 구성 데이터 세트를 보내고 받을 수 있습니다.

NETCONF 프로토콜은 **원격 프로시저 호출(RPC, Remote Procedure Call)** 패러다임을 사용합니다.

* 클라이언트는 XML 형식으로 RPC 요청을 인코딩하여 서버로 전송합니다.
* 서버는 XML 형식의 응답을 반환합니다.
* 요청 및 응답의 내용은 XML DTD(문서 타입 정의, Document Type Definition) 또는 XML 스키마로 완전히 기술되며, 이를 통해 통신하는 양측이 교환되는 데이터의 구문(syntax) 제약 조건을 인식할 수 있습니다.

**NETCONF의 핵심 기능**

1. **기능 확장 지원**
   * NETCONF는 장치의 기본 기능을 관리 프로토콜에서 그대로 활용할 수 있도록 하여, 구현 비용을 줄이고 새로운 기능에 신속하게 접근할 수 있도록 합니다.
   * 또한, 애플리케이션은 장치의 네이티브 사용자 인터페이스(UI)의 구문 및 의미적 내용에 액세스할 수 있습니다.
2. **서버의 기능(캡터빌리티, Capabilities) 탐색 가능**
   * NETCONF 클라이언트는 서버가 지원하는 확장 기능(캡터빌리티)을 검색할 수 있습니다.
   * 이를 통해 클라이언트는 서버에서 제공하는 특정 기능을 활용할 수 있도록 동작을 조정할 수 있습니다.
   * 표준 및 비표준 기능을 엄격한 의미적, 구문적 기준을 바탕으로 정의할 수 있습니다.
3. **자동화된 구성 시스템과의 연계 가능**
   * NETCONF는 자동화된 네트워크 구성 시스템의 중요한 구성 요소로 작동할 수 있습니다.
   * XML을 기본 데이터 표현 방식으로 사용하며, XSLT(XML 스타일시트 변환 언어) 같은 XML 변환 기술과 함께 활용될 수 있습니다.
   * 이를 통해 벤더 독립적인 구성 데이터 모델을 특정 벤더, 제품, 운영 체제(OS), 소프트웨어 릴리즈에 맞춰 변환하고, 변환된 데이터를 NETCONF 프로토콜을 통해 장치에 전달할 수 있습니다.

**RFC 2119 규칙**

이 문서에서 사용되는 다음 용어는 **RFC 2119**에 따라 해석해야 합니다.

* **MUST(반드시)**, **REQUIRED(필수적)**: 반드시 따라야 하는 요구 사항
* **SHALL(해야 한다)**, **SHALL NOT(해서는 안 된다)**: 엄격한 요구 사항
* **SHOULD(권장됨)**, **SHOULD NOT(권장되지 않음)**: 특정 상황에서 예외 가능
* **MAY(선택 사항)**, **OPTIONAL(선택적)**: 구현 여부가 자유로움

**설명 요약**

NETCONF는 네트워크 장치의 구성 및 관리를 위한 프로토콜로, XML 기반의 원격 프로시저 호출(RPC)을 사용하여 장치와 클라이언트 간의 통신을 수행합니다.

* 이 프로토콜을 사용하면 장치가 자체적인 API를 노출하고, 클라이언트는 이를 통해 설정을 가져오거나 변경할 수 있습니다.
* NETCONF는 기능 확장을 지원하며, 서버가 제공하는 기능을 동적으로 탐색할 수 있습니다.
* XML을 기반으로 하여 자동화된 네트워크 구성 시스템과 쉽게 연계될 수 있습니다.
* 문서 내에서 사용되는 주요 용어(MUST, SHALL, SHOULD 등)는 RFC 2119 규칙에 따라 엄격하게 해석되어야 합니다.

## 1.1 Terminology

**✅ 구성 데이터 저장소 (Configuration Datastore) 관련 용어**

1. **candidate configuration datastore (후보 구성 저장소)**
   * 현재 장치의 설정을 변경하지 않고 구성을 조작할 수 있는 저장소입니다.
   * 변경된 구성은 실행 구성 저장소(running configuration datastore)로 커밋(적용)할 수 있습니다.
   * 모든 장치가 이 기능을 지원하는 것은 아닙니다.
2. **running configuration datastore (실행 구성 저장소)**
   * 장치에서 현재 활성 상태인 구성 데이터를 저장하는 저장소입니다.
   * NETCONF에서는 항상 존재하는 필수적인 저장소입니다.
3. **startup configuration datastore (시작 구성 저장소)**
   * 장치가 부팅될 때 로드하는 초기 구성 데이터를 저장하는 저장소입니다.
   * 실행 구성 저장소와 별도로 운영되는 장치에서만 존재합니다.
4. **configuration datastore (구성 저장소)**
   * 장치를 초기 상태에서 원하는 작동 상태로 변환하는 데 필요한 **완전한 구성 데이터**를 보관하는 저장소입니다.
5. **datastore (데이터 저장소)**
   * 정보를 저장하고 액세스할 수 있는 **개념적인 저장 공간**을 의미합니다.
   * 예를 들어, 파일, 데이터베이스, 플래시 메모리 등의 형태로 구현될 수 있습니다.

**✅ 프로토콜 관련 용어**

1. **기능 확장 (capability)**
   * NETCONF의 기본 사양을 보완하는 추가적인 기능입니다.
2. **클라이언트 (client)**
   * 서버에서 제공하는 NETCONF 프로토콜 기능을 호출하는 역할을 합니다.
   * 또한, 서버에서 보내는 **알림(Notification)**을 구독할 수도 있습니다.
3. **서버 (server)**
   * 클라이언트가 요청한 NETCONF 프로토콜 작업을 실행하는 역할을 합니다.
   * 클라이언트에게 **알림(Notification)**을 보낼 수도 있습니다.
4. **세션 (session)**
   * 클라이언트와 서버가 보안이 보장된 연결을 통해 메시지를 주고받는 과정입니다.
5. **프로토콜 작업 (protocol operation)**
   * NETCONF 프로토콜에서 수행하는 특정 원격 프로시저 호출(RPC)입니다.
6. **원격 프로시저 호출 (remote procedure call, RPC)**
   * 클라이언트가 <rpc> 요청을 보내고, 서버가 이에 대한 <rpc-reply> 응답을 반환하는 방식으로 동작합니다.

**✅ 데이터 관련 용어**

1. **구성 데이터 (configuration data)**
   * 장치를 기본 상태에서 원하는 현재 상태로 변환하는 데 필요한 **설정 가능한 데이터**입니다.
2. **상태 데이터 (state data)**
   * 장치의 **운영 상태 정보를 나타내는 데이터**로, 예를 들어 읽기 전용의 장치 상태나 통계 데이터가 포함됩니다.
   * 구성 데이터와 달리, 사용자가 직접 수정할 수 없습니다.
3. **메시지 (message)**
   * 세션을 통해 전송되는 **NETCONF 프로토콜 요소**를 의미합니다.
   * 모든 메시지는 **XML 형식의 문서**로 구성됩니다.
4. **알림 (notification)**
   * 서버가 특정 이벤트를 감지했을 때 클라이언트에게 보내는 **서버 주도 메시지**입니다.

**✅ 사용자 관련 용어**

1. **사용자 (user)**
   * 클라이언트의 **인증된 ID**를 의미하며, 일반적으로 **NETCONF 사용자 이름**으로 표현됩니다.

**📌 요약**

NETCONF 프로토콜은 **구성 데이터(Configuration Data)와 상태 데이터(State Data)를 저장하는 다양한 저장소(Datastore)**를 사용하여 네트워크 장치를 관리합니다.

* **후보 구성 저장소 (Candidate Configuration Datastore)**: 변경 사항을 적용하기 전에 저장하는 임시 저장소 (모든 장치에서 지원하지는 않음).
* **실행 구성 저장소 (Running Configuration Datastore)**: 현재 실행 중인 구성 정보를 저장하는 저장소.
* **시작 구성 저장소 (Startup Configuration Datastore)**: 장치가 부팅될 때 로드하는 초기 구성 정보를 저장하는 저장소.
* **구성 데이터 (Configuration Data)**: 장치 설정을 변경하는 데이터.
* **상태 데이터 (State Data)**: 읽기 전용 정보 및 통계 데이터.

NETCONF는 **클라이언트-서버 모델**을 기반으로 작동하며, 클라이언트가 **RPC (Remote Procedure Call)** 요청을 보내면 서버가 이를 처리하고 응답합니다.

* **클라이언트(Client)**: 서버의 기능을 호출하고, 알림(Notification)을 받을 수 있음.
* **서버(Server)**: 클라이언트의 요청을 실행하고, 알림을 전송할 수 있음.
* **세션(Session)**: 클라이언트와 서버 간의 보안 연결.

## 1.2 Protocol Overview

NETCONF는 **RPC(Remote Procedure Call) 기반 통신 방식**을 사용하여 클라이언트와 서버 간의 상호작용을 단순화합니다.

* **클라이언트 (Client)**
  + 네트워크 관리자의 **스크립트 또는 애플리케이션**이 될 수 있습니다.
  + 일반적으로 네트워크 관리 소프트웨어의 일부로 실행됩니다.
* **서버 (Server)**
  + 네트워크 장치(Network Device)입니다.
  + 문서에서는 "서버"와 "장치"를 같은 의미로 사용합니다.
  + 클라이언트의 요청을 처리하고 응답을 반환합니다.

**✅ NETCONF 세션 (Session)**

* **NETCONF 세션**은 **네트워크 관리자 또는 네트워크 구성 애플리케이션과 네트워크 장치 간의 논리적 연결**입니다.
* 모든 장치는 **최소한 하나 이상의 NETCONF 세션을 지원해야 하며(MUST), 여러 개의 세션을 지원하는 것이 권장됩니다(SHOULD).**
* **세션의 속성**
  + **전역 속성(Global Attributes)**: 모든 세션에서 변경 사항이 즉시 반영됨.
  + **세션별 속성(Session-Specific Attributes)**: 해당 세션에서만 적용됨.

**✅ NETCONF 프로토콜 계층 구조**

NETCONF는 개념적으로 **4가지 계층**으로 나뉘며, 이는 **아래 그림** 과 같이 표현할 수 있습니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**(1) 보안 전송 계층 (Secure Transport Layer)**

* **클라이언트와 서버 간의 안전한 통신 경로를 제공**합니다.
* NETCONF는 여러 보안 전송 프로토콜 위에서 동작할 수 있습니다.
  + 예) **SSH, TLS, BEEP/TLS, SOAP/HTTP/TLS** 등.
* NETCONF에서 요구하는 기본적인 전송 프로토콜 요건은 **2장에서 설명**됩니다.

**(2) 메시지 계층 (Messages Layer)**

* **RPC 요청과 응답, 알림(Notification)을 포함하는 메시지를 처리**합니다.
* 메시지는 **전송 프로토콜에 독립적인 방식**으로 구성됩니다.
* 관련 내용은 **4장에서 설명되며,** [**RFC5717**](https://datatracker.ietf.org/doc/html/rfc5717)**에서 알림(Notification) 메시지를 정의**합니다.

**(3) 작업 계층 (Operations Layer)**

* **NETCONF의 기본적인 프로토콜 작업(Operations)을 정의**합니다.
* XML을 이용하여 **RPC 메서드와 매개변수를 인코딩** 합니다.
* 주요 작업:
  + <edit-config>: 장치의 구성 데이터를 수정.
  + <get>: 장치에서 데이터를 조회.
  + <commit>: 후보 구성 저장소의 변경 사항을 실행 구성 저장소에 적용.
* **기본 프로토콜 작업 목록은 7장에서 자세히 설명됩니다.**

**(4) 콘텐츠 계층 (Content Layer)**

* NETCONF에서 **처리할 데이터(구성 데이터, 알림 데이터 등)**를 의미합니다.
* 이 문서의 범위를 벗어나지만, NETCONF 데이터를 표준화하기 위한 별도의 작업이 진행되고 있습니다.

**✅ YANG과 NETCONF**

* **YANG**은 NETCONF에서 사용할 **데이터 모델을 정의하는 언어**입니다.
* **YANG은 NETCONF의 "작업 계층(Operations Layer)"과 "콘텐츠 계층(Content Layer)"을 다룹니다.**
* [**RFC 6020**](https://datatracker.ietf.org/doc/html/rfc6020)에서 정의되었으며, **NETCONF의 데이터 모델링을 위한 표준**입니다.

**📌 요약**

NETCONF는 **RPC 기반 프로토콜**로, 네트워크 장치를 원격에서 구성하고 관리할 수 있도록 합니다.  
**4가지 계층 구조**를 기반으로 동작합니다.

1. **보안 전송 계층**: SSH, TLS 등 보안 통신을 제공.
2. **메시지 계층**: RPC 요청/응답과 알림 메시지를 처리.
3. **작업 계층**: <edit-config>, <get> 등 NETCONF의 주요 작업을 정의.
4. **콘텐츠 계층**: 구성 데이터 및 알림 데이터를 포함.

YANG은 NETCONF의 **데이터 모델을 정의하는 표준 언어**로, 주로 **작업 계층과 콘텐츠 계층을 지원**합니다.

## 1.3 Capabilities

**NETCONF의 기능 확장(Capability)이란?**

* **NETCONF 기본 사양을 보완하는 기능 집합**을 의미합니다.
* 기능 확장은 **URI(Uniform Resource Identifier,** [**RFC 3986**](https://datatracker.ietf.org/doc/html/rfc3986)**)에 의해 식별**됩니다.

**✅ 기능 확장의 역할**

기능 확장은 **장치의 기본 작업을 보완**하며, 다음과 같은 역할을 합니다.

1. **추가 작업(Operations) 제공**
   * 기본 NETCONF 명령 외에 추가적인 원격 프로시저 호출(RPC) 제공 가능.
   * 예: <validate>, <rollback-on-error> 등의 고급 작업.
2. **작업 내 허용되는 콘텐츠(Content) 정의**
   * 특정 작업에서 어떤 데이터나 매개변수가 허용되는지 결정.
3. **클라이언트가 서버 기능을 동적으로 인식(Discovery) 가능**
   * 클라이언트는 서버의 기능을 확인한 후, 해당 기능을 활용할 수 있음.

**✅ 기능 간의 종속성 (Capability Dependencies)**

* **일부 기능은 다른 기능을 필요로 할 수도 있음.**
* 특정 기능을 지원하려면 **해당 기능이 의존하는 모든 기능을 서버가 지원해야 함(MUST).**

**✅ 기능 교환 (Capabilities Exchange)**

* **클라이언트가 서버의 지원 기능을 확인하는 과정**입니다.
* **8장에서 기능 교환 방식과 NETCONF에서 정의된 기능 목록을 설명합니다.**

**✅ 기능 확장의 유연성**

NETCONF 기능 확장은 **확장성이 뛰어나며, 새로운 기능을 쉽게 추가할 수 있음.**

* **언제든지 외부 문서에서 새로운 기능을 정의할 수 있음.**
* **표준화 기구(Standards Bodies)**가 공식적인 기능을 정의할 수 있음.
* **벤더(Vendor)별 맞춤형 기능**도 정의 가능함.

⚠ **기능 URI는 중복되지 않도록 고유해야 함(MUST).**

* 이를 위해 **URI에 해당 기능을 정의한 기관(표준 기구 또는 벤더)의 명칭을 포함**해야 함.

**📌 요약**

* **NETCONF 기능 확장(Capability)** 은 **NETCONF의 기본 기능을 확장**하는 역할을 함.
* **추가 작업, 콘텐츠 정의, 클라이언트가 서버 기능을 인식하는 기능을 제공**함.
* 기능 간의 **종속성(Dependency)이 있을 수 있으며**, 서버는 **의존하는 모든 기능을 지원해야 함.**
* **클라이언트와 서버는 기능 교환(Capabilities Exchange)을 통해 지원 기능을 협상할 수 있음.**
* **새로운 기능은 표준 기구 또는 벤더에서 정의 가능하며, URI를 통해 고유하게 식별됨.**

## 1.4 Separation of Configuration and State Data

구성 데이터(Configuration Data)와 상태 데이터(State Data)의 구분

NETCONF 프로토콜에서는 **장치에서 검색할 수 있는 정보**를 두 가지 유형으로 구분합니다.

**✅ 1. 구성 데이터(Configuration Data)**

* **쓰기 가능한 데이터(Editable Data)**
* **장치가 초기 상태에서 현재 상태로 변경되는 데 필요한 정보**
* 예제: 네트워크 인터페이스 설정, 라우팅 테이블, 방화벽 규칙 등

**✅ 2. 상태 데이터(State Data)**

* **구성 데이터가 아닌 추가적인 정보**
* **읽기 전용(읽기만 가능, 수정 불가능) 상태 정보 및 통계 데이터 포함**
* 예제: CPU 사용량, 메모리 사용량, 패킷 전송량, 현재 네트워크 연결 상태 등

**✅ 왜 구성 데이터와 상태 데이터를 구분해야 할까?**

만약 NETCONF에서 **구성 데이터와 상태 데이터를 구분하지 않으면** 다음과 같은 문제가 발생할 수 있습니다.

1. **구성 데이터 비교 시 불필요한 요소 포함**
   * 예) 통계 데이터(트래픽 카운터, 메모리 사용량 등)가 포함되면 구성 변경 사항을 비교하기 어려워짐.
2. **읽기 전용 데이터를 변경하려는 비논리적 요청 발생**
   * 예) CPU 사용량을 수정하려는 요청은 의미가 없음.
3. **데이터 크기 증가로 인해 성능 저하 발생**
   * 상태 데이터가 포함되면 불필요하게 큰 데이터셋을 다뤄야 함.
4. **백업 및 복원 시 복잡성 증가**
   * 예) 백업 파일에 읽기 전용 데이터가 포함되면, 복원 시 불필요한 정보까지 복원해야 할 수도 있음.

**✅ NETCONF에서의 해결 방법**

NETCONF 프로토콜은 **구성 데이터와 상태 데이터를 별도로 처리할 수 있도록 설계됨**.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

📌 **즉, 상태 데이터가 필요하지 않다면 <get-config> 만 사용하면 됨.**  
📌 **장치의 전체 상태(구성 + 상태 데이터)를 조회하려면 <get> 사용.**

**✅ 구성 데이터 외의 기타 데이터는?**

* NETCONF는 기본적으로 **장치를 원하는 상태로 만들기 위한 정보에 초점을 맞춤**.
* 하지만, **장치에 영구적으로 저장되는 기타 데이터(예: 사용자 파일, 데이터베이스 등)**는 구성 데이터에 포함할지 여부가 **제조사 또는 벤더의 구현 방식에 따라 다름**.

📌 **예제:**

* 네트워크 장치에 **사용자 인증 데이터(예: 계정 정보, 비밀번호 해시 등)가 저장된 데이터베이스**가 있다면?
  + 어떤 장치는 이를 **구성 데이터로 취급하여 NETCONF를 통해 관리 가능**하도록 할 수도 있고,
  + 어떤 장치는 이를 **구성 데이터에서 제외하고 별도의 관리 방법을 제공할 수도 있음**.

**📌 요약**

✅ **구성 데이터 (Configuration Data)**: 장치를 원하는 상태로 변경하는 데 필요한 **쓰기 가능한 설정 데이터**  
✅ **상태 데이터 (State Data)**: **읽기 전용(변경 불가)** 상태 정보 및 통계 데이터

✅ **구성 데이터와 상태 데이터를 구분하는 이유:**

* 구성 변경 비교 시 **불필요한 데이터 방지**
* **읽기 전용 데이터를 변경하려는 비논리적 요청 방지**
* **데이터 크기 증가로 인한 성능 저하 방지**
* **백업 및 복원 시 복잡성 감소**

✅ **NETCONF는 두 개의 명령어로 데이터를 조회:**

* <get-config>: **구성 데이터만 조회**
* <get>: **구성 데이터 + 상태 데이터 조회**

✅ **구성 데이터 외의 추가적인 영구 데이터(예: 사용자 DB)는 벤더(제조사) 구현에 따라 포함될 수도 있고 아닐 수도 있음.**

# 2. Transport Protocol Requirements