01 용어 정리

Glossary

QoS 가이드를 읽다 보면 Check Point 특유의 용어가 자주 나옵니다. 여기서는 QoS를 이해하는 데 꼭 필요한 용어 만 골라 흐름에 따라 풀어 둡니다. 나머지 일반적인 Check Point 용어는 필요할 때 본문에서 다시 설명합니다.

QoS의 핵심 개념

가장 먼저 **QoS** 자체입니다. QoS는 Security Gateway 위에서 도는 Software Blade로, 정책 기반으로 트래픽 대역폭을 관리해 업무에 중요한 트래픽을 우선시하고, 대역폭을 보장하며 지연(latency)을 통제 합니다.

> *QoS — Check Point Software Blade on a Security Gateway that provides policy-based traffic bandwidth management to prioritize business-critical traffic and guarantee bandwidth and control latency.*
>
> — Glossary p.21

QoS가 대역폭을 나누는 방식의 뿌리에는 **WFQ(Weighted Fair Queuing)** 가 있습니다. QoS에서 대역폭 배분을 정밀하게 통제하는 알고리즘 으로, 한국어로는 "가중 공정 큐잉"입니다. 이 WFQ를 구현하는 엔진을 **Intelligent Queuing Engine** 이라 부르는데, 높은 우선순위 트래픽이 낮은 우선순위 트래픽보다 먼저 처리되도록 보장 하는 대역폭 할당 알고리즘입니다. 규칙(rule)에 매겨지는 **QoS Action Properties** 는 한 규칙에 대한 대역폭 할당·제한(limit)·보장(guarantee)을 정의 하는 속성을 말합니다.

지연에 민감한 음성·영상을 위해서는 **LLQ(Low Latency Queuing)** 가 등장합니다. 원래 Cisco가 만든 기법으로, 지연에 민감한 데이터(음성 등)를 다른 트래픽보다 먼저 큐에서 꺼내 보내 우선 대우 합니다. 버퍼 관리에는 **WFRED(Weighted Flow Random Early Drop)** 가 쓰여, 트래픽 상황에 맞춰 자동·동적으로 패킷 버퍼를 조절합니다. 또 LAN을 WAN에 연결할 때 생기는 TCP 재전송을 줄이는 **RDED(Retransmit Detect Early Drop)** 는, 같은 패킷이 큐에 여러 벌 쌓이는 것을 감지해 불필요한 재전송을 막 습니다.

네트워크 품질을 말할 때 자주 나오는 단어로 **Jitter(지터)** 가 있습니다. 받는 쪽에서 패킷 간 지연이 들쭉날쭉해지는 변동 을 뜻합니다. 보내는 쪽은 패킷을 일정 간격으로 보내지만, 혼잡·잘못된 큐잉·설정 오류 때문에 받는 쪽 간격이 흔들리는 것입니다. **Burstiness(버스티니스)** 는 데이터가 짧고 불규칙한 폭발(spurt)로 몰려 전송되는 성질 로, LAN 트래픽이 전형적으로 이렇습니다(스트리밍 데이터의 반대).

구성 요소와 환경

QoS는 다른 Check Point 제품과 같은 뼈대를 씁니다. **Security Gateway** 는 트래픽을 검사하고 보안 정책을 집행하는 전용 서버이고(R81부터는 VSX의 Virtual System도 포함), **Security Management Server** 는 객체와 정책을 관리하는 서버, **SmartConsole** 은 정책을 만들고 장비를 관리하는 GUI 애플리케이션입니다. 정책의 규칙 묶음 전체를 **Rule Base** 라 하고, 그 안의 개별 조건·동작 한 줄이 **Rule** 입니다.

성능을 끌어올리는 기술도 함께 알아두면 좋습니다. **SecureXL** 은 게이트웨이를 지나는 IPv4·IPv6 트래픽을 가속하고, **CoreXL** 은 여러 CPU 코어에 방화벽 인스턴스를 병렬로 돌려 멀티코어 성능을 활용합니다. 이 둘은 QoS 소개에서 정책 모드와 함께 다시 나옵니다.

가상화·확장 환경 용어로는 **VSX(Virtual System Extension)** 가 있습니다. 한 장비나 클러스터 위에 여러 가상 Security Gateway를 올리는 Check Point의 가상 네트워킹 솔루션으로, 물리 장비와 같은 기능을 가상으로 제공합니다. QoS는 VSX를 완전히 지원합니다.

02 QoS 소개

Introduction to QoS

QoS는 한정된 회선을 누구에게 얼마만큼 줄지 정책으로 정하는 Check Point의 대역폭 관리 솔루션입니다. 이 장은 QoS가 무엇을 해 주는지, 어떤 기술로 그것을 해내는지, 그리고 정책에는 어떤 종류가 있는지를 큰 그림으로 잡습니다.

> 참 고
>
> R81부터 "Security Gateway"라는 말은 VSX의 Virtual System도 포함합니다.

QoS가 해 주는 일

QoS는 **정책 기반 대역폭 관리(policy based bandwidth management)** 솔루션으로, 크게 세 가지를 할 수 있습니다. ERP·데이터베이스·웹 같은 업무 핵심 트래픽을 낮은 우선순위 트래픽보다 우선 시키고, VoIP·화상회의 같은 스트리밍 애플리케이션에 대역폭을 보장하고 지연을 통제 하며, 특정 직원에게는 원격 접속 중에도 보장되거나 우선되는 접근권 을 줍니다. QoS는 Security Gateway에 얹어 배포하며, 암호화 트래픽과 비암호화 트래픽 모두 에 동작합니다.

!QoS 기본 구성 *① SmartConsole ② Security Management Server ③ QoS 정책 ④ QoS Software Blade를 켠 Security Gateway ⑤ 인터넷 ⑥ 내부 네트워크*

동작의 바탕에는 Check Point의 특허 기술인 **Stateful Inspection** 이 있습니다. 모든 네트워크 트래픽의 상태 정보를 잡아 동적으로 갱신하고, 이 상태 정보로 트래픽을 서비스·애플리케이션별로 분류합니다. 분류가 끝나면 계층적 WFQ(Weighted Fair Queuing) 알고리즘 을 적용해 대역폭 배분을 정밀하게 통제합니다.

특징과 이점

QoS의 핵심 강점은 가중치(weight)·제한(limit)·보장(guarantee)으로 짜는 유연한 정책 입니다. 기본 정책을 만들고 거기에 고급 기능을 덧붙이는 식이죠. 무엇보다 방화벽과 같은 아키텍처·기술 컴포넌트를 공유 한다는 점이 큽니다. 덕분에 VPN·Firewall·QoS 장비를 따로 둘 필요 없이 한 게이트웨이에서 처리하고, 사용자가 정의한 네트워크 객체를 두 솔루션이 함께 씁니다. 성능 분석은 SmartConsole의 Logs & Events 뷰로 하고, 공인망에서 트래픽에 등급을 매기는 **DiffServ**, 지연에 민감한 음성·영상을 위한 **Low Latency Queuing(LLQ)** 도 정책에 통합되어 있습니다. **CoreXL·SecureXL** 가속과 IPv6, 그리고 VSX도 완전히 지원합니다.

정책 종류 — Express와 Recommended

이번 릴리스에는 두 가지 QoS 정책 종류가 있습니다. **Express** 는 기본 QoS 정책을 빠르게 만드는 모드이고, **Recommended** 는 QoS의 전체 기능을 쓰는 고급 정책 모드입니다.

둘의 가장 큰 차이는 **고급 기능의 유무** 입니다. 가중치(Weight)·규칙 단위 제한(Limit)·로깅·하드웨어 가속·고가용성/로드 셰어링·VSX 같은 기본기는 양쪽 다 됩니다. 반면 연결(per-connection) 단위의 보장·제한, LLQ, DiffServ, 서브 규칙(sub-rule), URI/DNS 문자열 매칭 같은 정교한 기능은 Recommended에서만 쓸 수 있습니다. 그래서 확신이 없으면 Express로 시작 했다가 고급 기능이 필요해지면 Recommended로 올리는 편이 권장됩니다 (Express→Recommended 전환은 되지만 반대는 안 됩니다 — FAQ 참고).

정책 종류는 SmartConsole 메뉴에서 **Manage policies and layers** 를 열고, 새 정책을 만들거나 기존 정책을 편집하면서 **QoS** 를 선택한 뒤 **Recommended** 또는 **Express** 를 고르면 됩니다. 가중치·제한·보장의 자세한 동작은 기본 정책 관리에서 다룹니다.

R77 정책의 가속 지원

R82로 새로 설치하거나 업그레이드하면 QoS는 SecureXL·CoreXL 가속을 기본으로 켠 상태가 됩니다.

> 중 요
>
> 깨끗한 설치나 업그레이드 후 SecureXL·CoreXL은 기본으로 켜집니다. R77 이전용으로 만든 QoS 정책이라면, 가속이 켜진 상태에서 IPSO·R77.10 미만 게이트웨이·SmartView Monitor의 QoS 뷰 같은 기능이 지원되지 않습니다.

이런 옛 기능을 그대로 써야 한다면 가속을 꺼야 합니다(절차는 정규표현식 부록의 "QoS 가속 끄기" 참고).

작업 흐름

좋은 QoS 정책을 만드는 큰 흐름은 게이트웨이에서 QoS 켜기 → 전역 속성 설정 → 정책 만들기 → 로그/모니터링 설정 → 세션 게시(Publish) → 정책 설치 순입니다. SmartConsole에서 게이트웨이마다 QoS를 켜고 전역 속성을 잡은 뒤, (Legacy) SmartDashboard에서 실제 규칙과 DiffServ·LLQ 같은 특수 기능을 정의하고, 다시 SmartConsole로 돌아와 게시하고 설치합니다.

> 참 고
>
> QoS는 게이트웨이의 최소 한 개 인터페이스 에서 켜져 있어야 합니다. 어떤 인터페이스에서도 켜져 있지 않으면 정책 설치가 실패합니다. 그리고 Install Policy 창에서 반드시 **QoS** 를 선택해야 합니다.

구체적인 켜기 절차는 시작하기에서 단계별로 이어집니다.

한계

Scalable Platforms(Maestro·Chassis)에는 두 가지 제약이 있습니다. Security Group이 Layer 4 분산 모드일 때는 QoS를 지원하지 않 고, QoS 정책은 각 Security Group Member마다 개별로 적용 됩니다.

03 시작하기

Getting Started

QoS를 처음 켜는 일은 네 부분으로 나뉜 한 줄기 흐름 입니다. 게이트웨이 객체에서 QoS를 켜고(Part 1), 정책 패키지에서 QoS를 켜고(Part 2), 전역 파라미터를 잡고(Part 3), 실제 QoS 정책을 구성(Part 4)합니다. 이 장은 그 네 단계를 순서대로 따라갑니다.

Part 1 — 게이트웨이/클러스터 객체에서 QoS 켜기

먼저 SmartConsole로 Security Management Server(또는 Domain Management Server)에 접속합니다. 왼쪽 **Gateways & Servers** 에서 대상 Security Gateway 또는 Cluster 객체를 더블클릭해 엽니다.

객체 안에서는 두 곳 을 건드립니다. 먼저 **General Properties** 의 **Network Security** 탭에서 **QoS** 를 선택해 블레이드 자체를 켭니다. 그다음 **Network Management** 에서 대상 인터페이스를 골라 편집하고, 그 인터페이스의 **QoS** 항목에서 대역폭(Bandwidth)·DiffServ·Low Latency 클래스 등 필요한 설정을 잡습니다. 다 됐으면 인터페이스 창과 객체 창을 차례로 닫습니다. 인터페이스별 QoS 속성의 자세한 내용은 QoS 관리에서 다룹니다.

Part 2 — 정책 패키지에서 QoS 켜기

왼쪽 위 **Menu** > **Management policies and layers** 를 엽니다. 새 정책 패키지를 만들거나 기존 것을 편집한 뒤, **General** 의 **Policy Types** 에서 **QoS** 를 선택합니다. 이때 QoS 행의 **Mode** 에서 Recommended 또는 Express 를 고르는데, 두 모드의 차이는 QoS 소개에서 본 그대로입니다. 정책 창을 닫고 관리 창도 닫습니다.

Part 3 — QoS 전역 파라미터 설정

왼쪽 위 **Menu** > **Global properties** 를 열고 왼쪽에서 **QoS** 를 선택합니다. 여기서 규칙의 기본 가중치, 최대 가중치, 전송률 단위(예: Bps) 같은 전역 기본값을 잡습니다. 이 값들의 의미는 QoS 관리의 전역 속성 절에서 자세히 설명합니다.

Part 4 — QoS 정책 구성

이제 실제 규칙을 만듭니다. 왼쪽 **Security Policies** 로 가서 **Access Control** 의 **QoS** 를 클릭하고, **Open QoS Policy in SmartDashboard** 를 누릅니다. 그러면 Legacy SmartDashboard가 QoS 탭으로 열 립니다. 여기서 필요한 규칙을 구성한 뒤, **Menu** > **File** > **Update** 로 저장하고 **Menu** > **File** > **Exit** 로 빠져나옵니다. 마지막으로 Security Gateway/Cluster 객체에 Access Control 정책을 설치합니다.

> 참 고
>
> Install Policy 창에서 반드시 **QoS 정책** 을 선택해야 합니다.

규칙을 실제로 짜는 방법은 QoS 튜토리얼에서 예제로 따라가고, 규칙의 동작 원리는 기본 정책 관리에서 정리합니다.

04 QoS 배포

QoS Deployment

QoS를 어디에 어떻게 놓느냐가 효과를 좌우합니다. 이 장은 QoS가 트래픽을 제대로 통제하려면 배선이 어떻게 되어야 하는지 와, 실제 회선에 대역폭을 어떻게 나눠 주는지를 예제로 보여 줍니다.

배포의 두 가지 원칙

QoS는 게이트웨이가 지원하는 최대 인터페이스 수까지 관리할 수 있지만, 두 가지 전제 를 지켜야 합니다. 첫째, 관리 대상 회선의 모든 트래픽이 게이트웨이를 반드시 지나가야 합니다. 둘째, 관리 대상 회선은 각각 QoS 장비의 별도 물리 인터페이스에 (직접 또는 라우터를 거쳐) 연결 되어야 합니다. 두 회선이 한 물리 인터페이스를 공유하거나, 두 네트워크 구간이 같은 라우터에 붙으면 안 됩니다.

이유는 단순합니다. 게이트웨이가 보지 못하는 경로로 트래픽이 새면 통제할 수 없기 때문입니다.

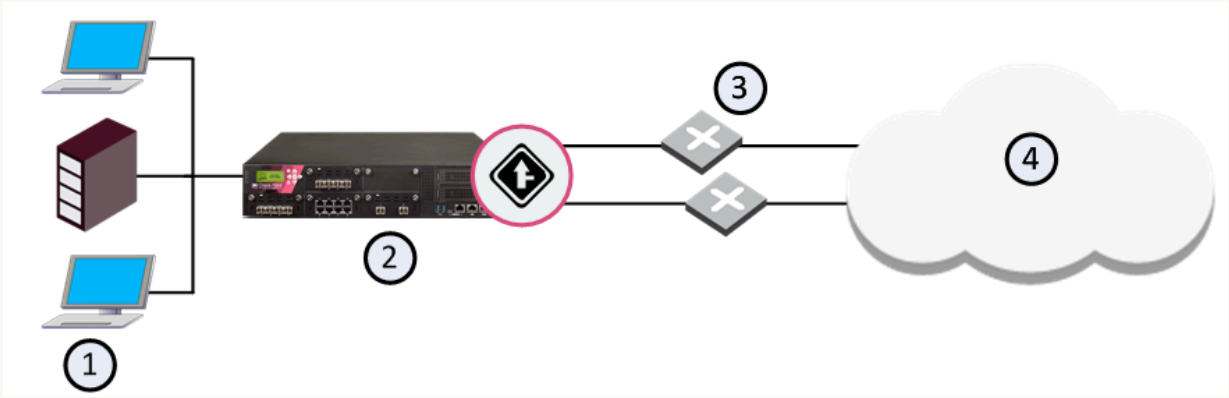
!잘못된 구성 — 허브로 우회 *① 내부 네트워크 ② QoS를 켠 Security Gateway ③ 허브 ④ 라우터들 ⑤ 인터넷*

위 그림처럼 라우터들이 허브를 통해 서로 트래픽을 주고받으면, QoS 게이트웨이는 그 트래픽을 알지 못합니다. 한 라우터에 두 네트워크가 붙어 있으면 트래픽이 라우터 안에서 바로 오가 버려 관리가 불가능합니다.

!올바른 구성 — 라우터가 게이트웨이에 직접 연결 *① 내부 네트워크 ② QoS를 켠 Security Gateway ③ 라우터들 ④ 인터넷*

올바른 구성에서는 라우터가 QoS 게이트웨이에 직접 연결 되어, 모든 트래픽이 게이트웨이를 거칩니다.

대역폭 배분 예제 — Frame Relay 망



Frame Relay 망 예제

!지점-본사 구성 *① 데이터베이스 서버 ② 웹 서버 ③ QoS를 켠 Security Gateway ④ 인터넷 ⑤ 지점(branch office)*

이 예제의 상황은 이렇습니다. 지점은 본사를 통해서만 인터넷과 직접 통신하고, 웹 서버는 지점에 회사 문서를 제공하며, 데이터베이스 서버는 회사의 핵심 업무를 받쳐 줍니다. 문제는 지점 트래픽 대부분이 내·외부 웹 트래픽이라, 정작 중요한 데이터베이스 트래픽이 밀린다 는 것입니다. 56K 회선을 증설하는 방법도 있지만, 비용도 비용이거니와 늘린 대역폭마저 웹 트래픽이 차지할 것 이라 근본 해결이 안 됩니다.

그래서 목표는 데이터베이스 서버 접속에 가장 큰 몫을 배정 하되, 지점이 56K 회선으로 연결되어 있다는 점 을 함께 고려하는 것입니다. 이를 다음과 같은 Rule Base로 달성합니다.

메인 규칙은 각 지점(Office 1 … Office n)에 가중치 10, 제한 56KBps 를 주고, 마지막 Default 규칙에 가중치 10을 둡니다. 그리고 각 지점 안에 서브 규칙 을 두어, 데이터베이스 트래픽에 가중치 50 을 주고 웹·기타 트래픽에는 10을 줍니다. 이렇게 하면 데이터베이스가 웹보다 우선 하게 됩니다.

구분	규칙	Source	Destination	Service	Action
메인	Office 1	Office 1	Any	Any	Weight 10, Limit 56KBps
메인	Default	Any	Any	Any	Weight 10
서브	Database Rule	Any	DB 서버	DB 서비스	Weight 50
서브	Web Rule	Any	Web 서버	http	Weight 10

전제 조건

이 예제에는 몇 가지 전제가 깔려 있습니다. 문제와 해법은 본사에서 나가는(outbound) 트래픽 에 적용되고, 본사는 최대 트래픽 부하를 감당할 용량 이 있으며, 지점 간 직접 트래픽은 없 다는 것입니다.

> 참 고
>
> QoS는 지점 회선을 나가는 방향(outbound)으로만 셰이핑합니다. 들어오는(inbound) 트래픽은 QoS 장비가 직접 통제하는 인터페이스 에서만 셰이핑됩니다.

가중치·제한·보장이 실제로 어떻게 대역폭을 나누는지는 QoS 튜토리얼과 기본 정책 관리에서 숫자로 따져 봅니다.

05 QoS 기본 아키텍처

Basic QoS Architecture

QoS의 내부 구조와 흐름 제어는 방화벽과 거의 똑같 습니다. 이 장은 QoS가 어떤 컴포넌트로 이뤄지고, 패킷이 어떤 경로로 처리되며, 방화벽·VPN과 어떻게 맞물리는지를 정리합니다.

세 컴포넌트

QoS는 방화벽처럼 세 컴포넌트 로 나뉩니다 — **SmartConsole**, **Security Management Server**, **Gateway**. 이들은 한 장비에 다 올릴 수도 있고, 여러 장비에 나눠 배치(distributed)할 수도 있습니다.

역할 분담은 이렇습니다. 대역폭 정책은 SmartConsole에서 구성 하고, Security Management Server에서 정책을 검증해 QoS 게이트웨이에 설치 합니다. 실제 트래픽을 다루는 QoS Security Gateway는 방화벽의 체이닝(chaining) 메커니즘으로 패킷을 받고·처리하고·보내 며, 자체 분류·규칙 매칭 인프라로 패킷을 검사 합니다. 로그 정보는 방화벽 커널 API로 만들어집니다.

QoS Blade — 커널 드라이버와 데몬

QoS Blade 의 본래 임무는 네트워크 접점에서 QoS 정책을 집행 하고 들어오고 나가는 트래픽의 흐름을 통제 하는 것입니다. 이 블레이드는 QoS 커널 드라이버 와 QoS 데몬 두 부분으로 이뤄집니다.

이 중 **QoS Engine** 이 QoS 동작의 심장 입니다. Firewall-1과 SecureXL의 일부로서, IP 패킷을 검사하고 큐에 넣고 스케줄링해 내보내는 일을 하며, 바로 이 과정이 QoS의 트래픽 통제를 가능하게 합니다.

사용자 모드 프로세스인 **QoS 데몬(fgd50)** 은 두 가지를 맡습니다. 커널을 위해 DNS를 해석해 Rule Base 매칭에 쓰게 하고, Cluster Load Sharing 구성에서는 클러스터 상태 변화를 커널에 알 립니다. 예를 들어 클러스터 멤버 하나가 죽으면, 데몬이 게이트웨이들의 상대 부하를 다시 계산해 커널을 갱신합니다.

SmartConsole과 SmartDashboard

정책을 짜는 도구는 둘로 나뉩니다. SmartConsole로 QoS 정책을 만들고 바꾸 며, 활성 게이트웨이와 정책 정보는 Logs & Events로 봅니다. 한편 실제 규칙과 거기 딸린 네트워크 객체·서비스 작업은 SmartDashboard 에서 합니다. QoS 정책의 규칙들은 **QoS Rule Base** 에 나타납니다.

QoS 구성 — 분산 배치

Security Management Server와 QoS Security Gateway는 한 장비에 둘 수도, 두 장비에 나눠 둘 수도 있습니다. 따로 두는 구성을 분산(distributed) 이라 부릅니다.

!분산 구성 *① 내부 네트워크(본사) ② QoS를 켠 Security Gateway ③ Security Management Server ④ SmartConsole ⑤ 인터넷 ⑥ QoS를 켠 Security Gateway(지점) ⑦ 내부 네트워크(지점)*

이 예에서는 하나의 Security Management Server가 네 대의 QoS 게이트웨이를 제어 하고, 그 게이트웨이들이 세 개의 QoS 회선에서 대역폭을 관리합니다. 한 Management Server가 여러 QoS 게이트웨이를 제어·모니터링 할 수 있고, 각 QoS Security Gateway는 Management Server와 독립적으로 동작합니다.

Client-Server 구조

SmartConsole과 Security Management Server도 같은 장비에 둘 수도, 나눌 수도 있습니다. 나누면 SmartConsole(클라이언트)이 Security Management Server(서버)를 제어하는 Client/Server 모델 이 됩니다.

!Client-Server 구성 *① 내부 네트워크(본사) ② Security Management Server ③ SmartConsole ④ QoS를 켠 Security Gateway ⑤ 인터넷*

이 그림에서는 Management Server의 기능이 두 워크스테이션(Tower와 Bridge)으로 나뉘어, 데이터베이스를 가진 Management Server는 Tower에, SmartConsole은 Bridge에 있습니다. Bridge에서 작업하는 사용자가 Tower에 있는 QoS 정책·데이터베이스를 유지하고, London의 QoS 게이트웨이가 회선에 정책을 집행합니다.

Management Server는 `cpstart` 명령으로 시작하며, 클라이언트에서 SmartConsole을 쓰려면 이 서버가 떠 있어야 합니다. 또 SmartConsole에 로그인한 관리자와 그 PC가 모두 서버 접근 권한을 받아야 서버를 관리할 수 있습니다. 권한 부여와 관리자 정의는 `cpconfig` 로 합니다.

동시 세션

여러 관리자가 동시에 각자 다른 세션에서 QoS 정책을 작업 할 수 있습니다. 다만 잠금(locking) 메커니즘이 같은 객체를 둘이 동시에 건드리지 못하게 막습니다. 작업을 마치고 **Publish** 를 누르면 변경 내용이 다른 세션·관리자에게 공개됩니다.

방화벽·VPN과의 상호작용

QoS와 방화벽은 많은 핵심 기술 컴포넌트를 공유 합니다. 같은 사용자 정의 네트워크 객체를 양쪽에서 쓰고, 효율적 검사와 성능을 위해 상태 테이블 정보를 함께 씁니다. 덕분에 암호화 트래픽과 NAT된 트래픽에도 대역폭 규칙 을 정의할 수 있습니다.

또 QoS는 방화벽과 객체 데이터베이스(네트워크 객체·서비스·리소스)를 공유 합니다. 다만 객체 중에는 제품별 고유 속성도 있습니다. 예를 들어 방화벽의 암호화 속성은 QoS와 무관하고, QoS 네트워크 인터페이스의 속도 속성은 방화벽과 무관합니다.

06 QoS 튜토리얼

QoS Tutorial

이 장은 예제 한 건을 처음부터 끝까지 따라가며 QoS 정책을 만들어 보는 실습입니다. 앞 장들의 개념(게이트웨이·Rule Base·SmartConsole/SmartDashboard·방화벽 블레이드)을 어느 정도 알고 있다는 전제로 진행합니다.

이 튜토리얼의 배포 시나리오

!튜토리얼 시나리오 *① Oxford — Security Management Server ② Cambridge — SmartConsole 클라이언트 ③ LAN(Engineering·Marketing) ④ London — QoS Security Gateway (4a eth2 199.199.199.32 / 4b eth1 199.32.43.32 / 4c eth0 199.32.32.32) ⑤ Web·FTP 서버가 있는 DMZ ⑥ 인터넷*

시나리오는 London·Oxford·Cambridge 세 곳에 사무실을 둔 조직입니다. QoS 게이트웨이는 London 에 있고 인터페이스가 셋(하나는 인터넷 연결), Management Server는 Oxford, SmartConsole은 Cambridge 에 있습니다. 내부망에는 Marketing과 Engineering 부서가 있습니다.

전체 흐름

튜토리얼은 구성요소 설치·설정 → 새 QoS 정책 생성 → 정책 종류 선택 → 네트워크 객체 구성 → 서비스 구성 → 규칙 생성 → 게이트웨이에 정책 설치 순으로 흘러갑니다.

구성요소 설치

먼저 London 게이트웨이에 QoS·Firewall 등 블레이드를 켜 고, Oxford에 Management Server를, Cambridge PC에 SmartConsole을 설치합니다. 그다음 SmartConsole에서 Cambridge를 신뢰 클라이언트로 지정 하고, QoS 정책을 관리할 관리자를 정의하며, Oxford와 London 사이에 SIC 신뢰 가 맺어졌는지 확인합니다.

새 QoS 정책 만들기

게이트웨이에서 QoS 블레이드가 켜져 있는지 확인한 뒤, SmartConsole의 **File** > **Manage Policies and Layers** 에서 **New** 를 누릅니다. 정책 이름을 정하는데, 예약어·공백을 쓰면 안 되고, 숫자로 시작하거나 `% # ' & * ! @ ? < > / \ :` 같은 문자를 넣으면 안 되며, `.pf` · `.W` 로 끝나도 안 됩니다. 그다음 **QoS** 를 선택하고 Express(기본 정책) 또는 Recommended(전체 기능, 기본값) 를 고릅니다. **OK** 를 누르면 정책이 저장되고 SmartDashboard가 자동으로 열려 규칙을 짜기 시작할 수 있습니다.

정책 설계와 게이트웨이 구성

좋은 정책을 짜려면 먼저 네트워크가 어떻게 쓰이는지 파악 해야 합니다. 트래픽 종류를 식별해 우선순위를 매기고, 사용자와 그 필요를 파악합니다. 이 예제의 방침은 HTTP는 RealAudio보다 많은 대역폭, Marketing은 Engineering보다 많은 대역폭 을 주는 것입니다.

게이트웨이 구성에서는 London 게이트웨이 객체와 Marketing·Engineering 서브넷 객체를 정의합니다. London 게이트웨이는 **Gateways & Servers** > **New** > **Gateway** > **Classic Mode** 로 만들고, 이름(London)·플랫폼·**SIC**(Communication 클릭)·버전(R82)·OS(Gaia)·IP를 채웁니다. 게이트웨이의 IP는 항상 외부 인터페이스 주소 를 씁니다. **Network Security** 탭에서 Firewall과 QoS를 켭니다.

인터페이스 정의

Network Management 에서 **Get Interfaces** 를 눌러 인터페이스를 가져온 뒤, 각 인터페이스(eth0/eth1/eth2)를 더블클릭해 주소·넷마스크·토폴로지·Anti-Spoofing을 설정합니다. 그리고 인터페이스 창의 **QoS 탭** 에서 Inbound Active·Outbound Active를 켜고, 둘 다 192000 — T1(1.5Mbps)로 전송률을 잡습니다.

서비스는 따로 만들 필요가 없습니다. 이 예제가 쓰는 HTTP와 RealAudio는 QoS에 이미 정의되어 있습니다.

규칙 만들기와 동작 이해

새 QoS 정책에는 항상 Default Rule이 자동으로 추가되며, 반드시 맨 마지막 에 있어야 합니다. 그러니 새 규칙은 모두 Default Rule **위** 에 넣습니다. 이 예제에서는 SmartDashboard의 QoS 탭에서 Default 규칙을 고르고 **Before current rule** 아이콘으로 Web Rule 과 RealAudio Rule 두 개를 만듭니다.

서비스로 분류하기

규칙의 기본값을 다음처럼 바꿉니다 — Web Rule은 HTTP에 Weight 35, RealAudio Rule은 RealAudio에 Weight 5, Default는 Weight 10. Service 필드를 우클릭해 객체를 더하고, Action 필드를 더블클릭해 Rule Weight를 고칩니다. DNS·ARP 같은 "배경" 서비스는 보통 명시하지 않고 Default 규칙이 처리 합니다.

연결은 자기에게 적용되는 규칙의 가중치(우선순위) 비율만큼 대역폭 을 받습니다. 네 연결이 동시에 열린 경우를 보면 — HTTP는 35/50 = 70%, RealAudio는 5/50 = 10%, FTP·TELNET은 Default 공유로 각각 10/50 = 20% 를 받습니다. 핵심은 대역폭 배분이 연결이 열리고 닫힐 때마다 계속 바뀐다 는 점입니다. HTTP·FTP·TELNET이 모두 닫히면 RealAudio가 100%를 받고, TELNET·FTP만 닫히면 HTTP와 RealAudio가 풀린 대역폭을 나눠 갖습니다. 가중치가 아무리 작아도 그 트래픽이 굶지는 않는다 는 것이 WFQ의 성질입니다(두 연결만 열리면 RealAudio도 5/40 = 12.5%는 받음).

출발지로 분류하기

방침의 둘째 항목(Marketing > Engineering)은 출발지 기반 규칙으로 구현합니다 — Marketing Rule에 Weight 30, Engineering Rule에 Weight 20, Default에 Weight 10. 동작 원리는 서비스 기반과 같고, 배분 기준이 서비스가 아니라 출발지 라는 점만 다릅니다.

First Rule Match 원칙

이제 모든 규칙을 한 Rule Base에 모으면, 한 연결이 여러 규칙에 걸릴 수 있습니다. QoS는 First Rule Match(가장 먼저 매칭된 규칙) 원칙 으로 동작합니다 — 위에서부터 검사해 처음 매칭된 규칙의 Action만 적용 합니다.

Source	Destination	Service	Action
Any	Any	HTTP	Weight 35 (Web Rule)
Any	Any	RealAudio	Weight 5
Marketing	Any	Any	Weight 30
Engineering	Any	Any	Weight 20
Any	Any	Any	Weight 10 (Default)

그래서 Marketing 사용자가 HTTP를 열면 Web Rule과 Marketing Rule 둘 다에 걸리지만, 위에 있는 Web Rule이 적용되어 Weight 35 를 받습니다. HTTP를 출발지별로 다시 나누고 싶으면 Web Rule 아래에 서브 규칙을 둡니다.

보장(Guarantee)·제한(Limit)과 서브 규칙

가중치 말고 보장과 제한 으로도 대역폭을 정할 수 있습니다. Web Rule이 "가용 대역폭의 35%"를 받는다는 건 총 대역폭과 다른 규칙의 열린 연결 수에 따라 실제 양이 달라진다 는 뜻인데, 보장은 이를 절대값(Bytes/초)으로 못 박 습니다. 예를 들어 Web Rule에 Guarantee 20KBps 를 주면, 거기 매칭된 연결은 총 20KBps를 받고 남는 대역폭을 가중치대로 다시 나눠 갖습니다.

서브 규칙(Sub-Rule) 은 규칙 안에 중첩된 규칙 입니다. 예컨대 Web Rule(Weight 20) 아래에 Marketing HTTP(Weight 10)와 Default(Weight 1) 서브 규칙을 두면, Web Rule의 대역폭이 서브 규칙들에 10:1 비율 로 나뉩니다. 이때 서브 규칙의 Source·Destination·Service는 부모 규칙의 부분집합 이어야 하고, 서브 규칙 묶음에도 자체 Default가 자동 으로 생깁니다. 서브 규칙은 규칙 이름을 우클릭해 **Add Sub-Rule** 로 만듭니다.

QoS 정책 설치

규칙을 다 짰으면 설치합니다. SmartDashboard에서 **Update** 로 변경을 반영하고, SmartConsole에서 **Install Policy** 를 누른 뒤, 설치할 정책을 고르고 **Policy Targets** 에서 대상 게이트웨이를 선택해 **Install** 합니다.

> 참 고
>
> QoS는 기본적으로 어떤 게이트웨이도 선택되어 있지 않 습니다. 반드시 직접 골라야 합니다.

설치가 실패하면 SmartConsole 왼쪽 아래 **Task Information** → **Recent Tasks** 에서 해당 작업의 **Details** 를 열고, Status 열의 `^` 아이콘으로 오류 메시지를 봅니다. 모든 오류를 해결해야 정책이 설치 됩니다.

07 기본 정책 관리

Basic Policy Management

이 장은 동작하는 QoS Rule Base를 정의하고 운영하는 데 필요한 기본기 를 정리합니다. 더 정교한 기능은 고급 정책 관리에서 이어집니다.

GUI 클라이언트 열기

SmartConsole은 Windows 시작 메뉴에서 엽니다. **SmartDashboard** 는 QoS 정책을 만들거나 기존 정책을 열면 자동으로 뜨므로 따로 열 일이 드뭅니다. 굳이 직접 열려면 SmartConsole에서 QoS 정책을 연 뒤 **Security Policies** > **Access Control** > **QoS** 로 가서 **Open QoS Policy in SmartDashboard** 를 누릅니다.

> 중 요
>
> Legacy SmartDashboard는 읽기 전용 권한 관리자가 로그인했고 정책 패키지에 "Desktop Security"가 켜져 있으면 QoS·Desktop 정책을 보여 주지 않습니다.

Rule Base의 동작 원리

QoS 정책은 Rule Base에 규칙을 정의해 구현 합니다. Rule Base는 패킷을 어떻게 다룰지 — 트래픽의 출발지·목적지, 쓸 수 있는 서비스, 시간, 로깅 여부와 수준 을 정합니다. 규칙은 우리가 만든 것들과 자동으로 생기는 Default Rule 로 이뤄지는데, Default는 수정은 되지만 삭제는 안 되고 항상 맨 마지막 에 있으며, 다른 규칙에 안 걸린 모든 패킷에 적용됩니다.

검사 방식은 순차적 입니다. QoS는 패킷을 첫 규칙부터 차례로 비교하다가 매칭되는 규칙을 찾으면 멈추고 그 규칙을 적용 합니다. 매칭된 규칙에 서브 규칙이 있으면 다시 서브 규칙들을 차례로 검사하고, 아무것도 안 맞으면 Default(또는 서브 Default)를 적용합니다. 가장 잘 맞는 규칙이 아니라, 가장 먼저 맞는 규칙 이 적용된다는 점이 핵심입니다.

> 권 장
>
> 규칙은 실제 트래픽 패턴에 근거 해 만드세요. Logs & Events로 트래픽 로그를 분석하면 좋습니다. 또 객체(네트워크 객체·서비스)는 그룹으로 묶으면 정책을 한눈에 보기 좋고 Rule Base가 읽기 쉬워지며, 그룹에 새 객체를 더하면 규칙에 자동 포함됩니다.

> 참 고
>
> R82부터는 목적지 포트가 같고 출발지 포트가 다른 서로 다른 Service 객체 도 QoS 정책에서 지원합니다.

연결 분류 — 네 가지 기준

한 연결은 네 가지 기준 으로 분류됩니다 — **Source**(출발지), **Destination**(목적지), **Service**(서비스), **Time**(시간). Source·Destination은 특정 컴퓨터·네트워크·사용자 그룹·도메인 같은 네트워크 객체 묶음이고, Service는 TCP·UDP·ICMP·URL 같은 IP 서비스 묶음이며, Time은 특정 요일·시간대입니다.

여기 쓰이는 객체들은 종류가 다양합니다. **네트워크 객체** 는 워크스테이션·네트워크·도메인·그룹을 포함하고, **사용자 그룹(User Group)** 으로 예컨대 마케팅 부서 사용자를 묶어 규칙의 Source로 쓸 수 있습니다. **서비스·리소스** 는 TCP·Compound TCP·UDP·ICMP·IP 서비스를 쓰며, 리소스는 QoS에서 URI 타입만 됩니다. **시간 객체(Time Object)** 는 규칙이 언제 집행될지를 요일·날짜로 지정합니다.

대역폭 배분의 세 요소 — 가중치·보장·제한

규칙은 분류된 연결의 대역폭에 세 가지 요소 를 적용할 수 있습니다.

먼저 **가중치(Weight)** 는 한 규칙에 배정되는 가용 대역폭의 비율 입니다.
계산식은 간단합니다.

```
가중치 비율 = (이 규칙의 우선순위) / (열린 연결이 있는 모든 규칙의 우선순위 합)
```

예를 들어 이 규칙의 우선순위가 12이고, 현재 연결이 열린 모든 규칙의 우선순위 합이 120이면, 이 규칙의 연결들은 12/120 = 10% 를 받습니다. 다만 다른 규칙이 최대치를 안 쓰면 이 규칙이 그보다 더 받 을 수 있습니다. 즉 남는 대역폭은 회선이 놀지 않도록 나머지 규칙들이 상대 가중치대로 나눠 갖습니다.

다음으로 **보장(Guarantee)** 은 매칭된 연결에 최소 대역폭을 할당 합니다. 규칙 전체 연결의 합 에 보장하거나(연결이 많을수록 연결당 몫은 줄어듦), 연결마다 개별 로 보장할 수 있습니다. 가중치도 대역폭 몫을 보장하지만, 절대값으로 못 박는 건 보장뿐 입니다.

마지막으로 **제한(Limit)** 은 연결들이 받을 수 있는 최대 대역폭 입니다. 남는 대역폭이 있어도 이 점을 넘으면 더 주지 않 습니다. 제한도 규칙 전체 합 또는 개별 연결 단위로 정할 수 있습니다.

> 참 고
>
> 대역폭 배분은 고정이 아닙니다. 연결이 열리고 닫힐 때마다 QoS가 정책에 따라 끊임없이 재조정 합니다.

Default Rule과 Action 속성

Default Rule 은 각 Rule Base에 자동으로 추가되어 전역 속성에 정해진 가중치 를 받습니다. 가중치는 바꿀 수 있어도 삭제는 안 되며, 다른 규칙에 안 걸린 모든 연결에 적용됩니다. 서브 규칙 묶음에도 각각 Default가 자동으로 생깁니다.

규칙의 동작은 **QoS Action Properties** 창에서 정의하며, Action 타입은 둘입니다. **Simple** 은 양쪽(Recommended·Express) 모두 쓸 수 있고 — 암호화 트래픽만 적용·규칙 가중치·규칙 제한·규칙 보장을 다룹니다. **Advanced** 는 Recommended 전용 으로 — 연결 단위 보장·제한, 영구 연결 수, 추가 연결 허용 같은 정교한 속성을 더합니다.

VPN 트래픽을 매칭하는 규칙 예

여기서 VPN 트래픽이란 이 Security Gateway가 직접 암호화한 트래픽 을 말합니다(다른 제품이 미리 암호화해 들어온 건 IPSec 서비스로 매칭). Action Properties에서 **Apply rule only to encrypted traffic** 을 켜면 VPN 트래픽만 그 규칙에 걸립니다.

QoS가 First Rule Match로 동작하므로, VPN 규칙은 Rule Base 맨 위에 두 어야 합니다. 그러면 VPN 트래픽이 위 규칙에 먼저 걸리고, 그 아래 규칙들은 비-VPN 트래픽만 검사 하게 됩니다. VPN 규칙 아래에 서브 규칙을 두어 VPN 트래픽을 더 세분할 수도 있습니다.

대역폭 배분과 서브 규칙

연결이 서브 규칙을 가진 규칙에 매칭되면 서브 규칙들을 검사하고, 아무것도 안 맞으면 서브 Default를 적용합니다. 서브 규칙은 중첩(nesting) 도 되어, 서브 규칙이 또 서브 규칙을 가질 수 있습니다.

배분은 위에서 아래로(top/down) 흐릅니다. 그래서 서브 규칙은 부모 규칙보다 많은 대역폭을 줄 수 없 고, 중첩 서브 규칙도 마찬가지입니다. 또 규칙의 Guarantee는 그 서브 규칙의 Guarantee보다 같거나 커야 합니다. 보장과 제한이 얽힐 때의 정확한 규칙은 고급 정책 관리에서 예제로 따집니다.

정책 사용과 설치

규칙을 다 정의했으면 SmartConsole 세션을 게시(Publish)한 뒤 게이트웨이에 정책을 설치합니다. 설치 과정에서 규칙과 객체가 자동 검증되고, 오류가 있으면 Install Policy Details 탭에 메시지가 뜹니다. 설치가 성공하면 게이트웨이가 규칙을 집행합니다.

설치 절차는 QoS 튜토리얼·QoS 관리와 같습니다 — SmartDashboard에서 **Update**, SmartConsole에서 **Install Policy** → 정책 선택 → **Policy Targets** 에서 대상 선택 → **Install**. 기본적으로 어떤 게이트웨이도 선택되어 있지 않으니 직접 골라야 하고, 설치 전 게이트웨이에서 QoS 블레이드가 켜져 있는지 확인합니다.

08 고급 정책 관리

Advanced QoS Policy Management

이 장은 기본 정책 관리에서 잡은 기본 정책을 더 정교하게 다듬는 방법을 다룹니다. 보장·제한을 규칙·서브 규칙에 얽을 때 지켜야 할 규칙, 공인망 등급을 매기는 **DiffServ**, 지연을 통제하는 **LLQ** 가 핵심입니다.

보장과 제한 — 예제로 보는 규칙들

규칙·서브 규칙의 Action 속성이 대역폭 배분을 결정합니다. 여기서는 보장과 제한을 제대로 쓰는 원칙 을 예제로 따져 봅니다.

규칙 단위 보장(Per Rule Guarantee)

규칙에 배정되는 대역폭은 보장된 양 + 가중치에 따른 몫 입니다. 보장을 지키려고 보장분을 총 대역폭에서 먼저 떼어 두고, 남은 대역폭을 모든 규칙의 가중치대로 나눕니다.

예를 들어 회선이 190KBps, Rule A가 Guarantee 100KBps에 Weight 10, Rule B가 Weight 20이라면 — Rule A는 130KBps(보장 100 + (10/30)×(190-100)=30)를, Rule B는 60KBps((20/30)×90)를 받습니다.

서브 규칙과 얽힐 때는 몇 가지 반드시 지킬 제약 이 있습니다. 서브 규칙에 보장을 정하면 그 위 부모 규칙에도 보장이 있어야 하고, 서브 규칙의 보장은 부모 규칙의 보장보다 클 수 없습니다. 또 부모 규칙의 보장은 그 서브 규칙들의 보장 합보다 작으면 안 됩니다(예: 서브 A1·A2가 각 80이면 합 160 > 부모 A의 100이라 틀림 → A를 160 이상으로). 그리고 최상위 규칙들의 보장 합은 회선 용량의 90%를 넘으면 안 됩니다. 보장은 매칭되는 연결이 있을 때만 대역폭을 잡 고, 연결 속도가 보장보다 낮으면 남는 양은 다른 연결이 쓸 수 있 게 풀립니다.

여기에 한 가지 함정이 있습니다. 규칙의 가중치가 너무 낮으면, 그 규칙에 매칭된 연결이 받을 대역폭이 거의 없 어질 수 있습니다. 예컨대 Rule A가 Guarantee 100·Weight **1** 이면 거의 103KBps만 받고, 그 안의 서브 A1이 보장 100을 다 가져가면 서브 A2는 1.5KBps밖에 못 받 습니다.

연결 단위 보장(Per Connection Guarantee)

연결마다 보장하는 경우, **Accept additional connections** 를 켜면 보장 연결 수를 넘는 연결도 열리 며, 그 옆 칸이 비어 있으면 추가 연결은 규칙 가중치대로 대역폭을 받습니다. 서브 규칙의 연결 단위 보장은 부모 규칙의 그것보다 클 수 없 고, 서브 규칙에 연결 보장이 없으면 부모 규칙의 값을 물려받 습니다.

제한과, 보장-제한의 상호작용

규칙은 Rule Limit와 Per connection Limit를 둘 다 가질 수 있지만 연결 제한이 규칙 제한보다 크면 안 됩니다. 서브 규칙 모두에 제한이 있으면 부모 규칙의 제한은 서브 규칙 제한들의 합보다 클 수 없 습니다.

보장과 제한이 한 규칙에 같이 있으면, 제한이 보장보다 작으면 안 되고, 제한 = 보장이면 연결이 대역폭을 못 받는 상황 이 생길 수 있습니다.

> 예 – 대역폭을 못 받는 경우
>
> Rule A에 Guarantee 100·Limit 100을 두고, 그 서브 A1에 Guarantee 100을 두면, A1이 보장분을 다 쓸 때 같은 부모 아래 보장 없는 서브 규칙의 트래픽은 대역폭을 거의 못 받 습니다(A의 제한이 100이라 더 줄 여지가 없으므로).

Differentiated Services (DiffServ)

DiffServ 는 네트워크 트래픽에 종류·등급별로 다른 서비스를 주는 아키텍처 입니다. 기업망 안에서 패킷의 IP 헤더 TOS 바이트에 어떤 등급(QoS Class)에 속하는지 표시(marking) 해 두면, 공인망(public network)으로 나갔을 때 그 등급에 따라 우선권을 받습니다. DiffServ 표시는 공인망에서 의미가 있고 기업망 안에서는 아닙니다. 그러니 거치는 모든 공인망 구간이 그 표시를 인정 해야 제대로 동작합니다.

IPSec 패킷에 DiffServ 표시를 쓸 때는, `$FWDIR/conf/objects_5_0.c` 의 속성으로 헤더 간에 표시를 복사 할 수 있습니다. `:ipsec.copy_TOS_to_inner` 는 복호화/역캡슐화 후 IPSec 헤더의 표시를 IP 헤더로, `:ipsec.copy_TOS_to_outer` 는 캡슐화 후 IP 헤더의 표시를 IPSec 헤더로 복사합니다. 기본값은 다음과 같습니다.

```
:ipsec.copy_TOS_to_inner (false)
:ipsec.copy_TOS_to_outer (true)
```

DiffServ 규칙도 일반 규칙처럼 QoS Class와 함께 가중치를 갖지만, 그 가중치는 해당 클래스 규칙이 설치된 인터페이스에서만 적용 됩니다. 예컨대 FTP에 weight 50인 DiffServ 규칙은 그 QoS Class가 정의된 인터페이스에만 설치 되고, 다른 인터페이스를 지나는 FTP는 그 가중치를 못 받습니다. 모든 FTP에 가중치를 주려면 "Best Effort"(비-DiffServ 규칙) 아래에 규칙을 추가합니다. QoS Class는 인터페이스 속성의 QoS 탭에서 정의합니다(QoS 관리 참고). 참고로 QoS는 매칭된 패킷에 DiffServ 표시를 더하는 건 지원하지만, DiffServ 태깅을 보고 패킷을 매칭하는 건 지원하지 않 습니다.

Low Latency Queuing (LLQ)

웹의 대부분 TCP 트래픽에는 WFQ면 충분합니다. 패킷을 큐에 넣고 인터페이스 대역폭과 규칙 우선순위에 따라 내보내, 패킷을 떨구지 않는 대신 (때로 긴) 큐를 유지 합니다. 문제는 음성·영상처럼 지연을 일정 한계 안으로 묶어야 하는 트래픽 입니다. 긴 큐는 큰 지연을 부르니까요. 다행히 이런 스트림은 비트레이트가 알려진 경우가 많 아, 들어오는 만큼 바로 내보내면 큐가 거의 안 쌓여 지연이 무시할 만해집니다.

LLQ 는 바로 이런 "지연 민감" 애플리케이션을 위한 특수 Class of Service 를 만들게 해 줍니다. 이 클래스에는 허용 최대 지연(Maximum Delay)과 Constant Bit Rate(고정 비트레이트) 를 지정하고, QoS는 매칭 트래픽이 그 지연 한계 안에서 전달되도록 보장 합니다.

Low Latency 클래스의 동작

각 LLQ 클래스에는 활성 방향마다 고정 비트레이트와 최대 지연을 정합니다. QoS는 패킷이 최대 지연을 넘겼는지 검사해, 넘긴 패킷은 떨구고 아니면 고정 비트레이트로 전송 합니다. 고정 비트레이트가 도착 속도보다 작지 않으면 패킷은 안 떨어지 고, 도착 속도가 고정 비트레이트보다 높으면 초과분을 떨궈 최대 지연을 지킵니다.

클래스 우선순위와 비트레이트·지연 계산

보통은 한 LLQ 클래스로 충분하지만, 트래픽마다 다른 최대 지연이 필요하면 여러 클래스를 둡니다. 클래스는 (Expedited Forwarding 제외) 다섯 단계 우선순위 를 가지며, 최대 지연이 낮은 클래스가 더 높은 우선순위 를 받아야 합니다. 두 클래스의 패킷이 동시에 준비되면 높은 우선순위 패킷이 먼저 나가고, 낮은 쪽은 더 큰 지연을 겪기 때문입니다. 그래서 우선순위를 최대 지연 순으로 정하고, 우선순위 높은 순으로 클래스를 정의 하는 것이 권장됩니다.

고정 비트레이트 계산 에는 한계가 있습니다. 모든 LLQ 클래스의 고정 비트레이트 합은 총 대역폭의 20%를 넘을 수 없 습니다 — 그래야 Best Effort 트래픽이 큰 지터를 안 겪습니다. 값 자체는 한 애플리케이션 스트림의 비트레이트 × 동시에 열릴 것으로 예상하는 스트림 수 로 잡습니다. 예상보다 스트림이 많으면 드롭이 늘어나니, 드롭을 막으려면 동시 스트림 수를 제한합니다.

최대 지연 계산 은 스트림이 견딜 수 있는 최대 지연 과 QoS가 보장할 수 있는 최소 지연 사이에서 정합니다. 너무 작게 잡으면 불필요한 드롭 이 생깁니다 — 지연이 작을수록 큐가

짧아져, 전달 전에 패킷이 떨어질 수 있기 때문입니다. 상한은 (애플리케이션이 견디는 지연) − (외부 WAN이 더하는 지연) 으로 잡습니다(음성은 보통 전체 지연 150ms를 넘으면 사용자가 이상을 느낌). 하한은 버스트가 없으면 `[3 × 패킷크기] / 비트레이트` (패킷 3개를 큐에 담을 여유), 버스트가 있으면 `[(버스트크기+1) × 패킷크기] / 비트레이트` 입니다. 실제 값은 상·하한 사이로 잡되 어느 한쪽에 너무 붙이지 말 고, 버스트가 의심되면 상한 쪽에 가깝게 둡니다.

> 권 장
>
> LLQ 로그에 찍히는 기본 Class Maximum Delay 값 을 활용하세요. 먼저 올바른 고정 비트레이트를 잡고 최대 지연을 어림한 뒤, 로그가 알려 주는 값을 참고합니다.

고정 비트레이트 초과 막기

LLQ 클래스를 지나는 총 비트레이트가 고정 비트레이트를 넘으면 드롭이 생깁니다. 막으려면 그 클래스 아래에 규칙을 하나 두고, Per Connection Guarantee로 연결당 비트레이트를 정하고, Number of guaranteed connections로 허용 연결 수를 못 박 습니다. 이때 **Accept additional non-guaranteed connections** 는 끕니다.

LLQ를 인터페이스처럼 생각하기

LLQ 클래스를 활성화하려면 그 아래에 규칙을 최소 하나 둬야 합니다. 매칭된 트래픽은 클래스의 지연·고정 비트레이트 속성과 규칙 속성(가중치·보장·제한)을 함께 적용받습니다. LLQ 클래스를 별도 네트워크 인터페이스처럼 생각하면 쉽습니다 — 클래스 지연 안에서 고정 비트레이트로 패킷을 내보내고, 그 앞에서 규칙들이 상대 우선순위를 정하는 셈입니다.

> 참 고
>
> LLQ 클래스 아래에서 서브 규칙은 권장하지 않 습니다(드롭 패턴 계산이 어려워짐). 보장·제한도 같은 이유로 권장하지 않으나, 위에서 본 Per Connection Guarantee(드롭 방지용)는 예외 입니다.

언제 LLQ를 쓰나

LLQ는 지연이 중요하고 들어오는 스트림의 비트레이트를 알 때 씁니다(음성·영상 — 최대 지연과 고정 비트레이트를 둘 다 지정). 지연 통제는 중요한데 비트레이트를 모르는 경우 도

됩니다(예: Telnet — 고정 비트레이트를 높게, 최대 지연을 아주 크게(99999ms) 잡으면, 버스트 때 패킷을 떨구지 않고 큐에 담아 고정 비트레이트로 내보냄). 반대로 지연 통제가 그리 중요치 않은 대부분의 TCP(HTTP·FTP·SMTP)에는 가중치·제한·보장이 더 적합 합니다.

LLQ vs DiffServ

LLQ 클래스는 TOS 표시를 받지 않 아, 우선 대우가 QoS 게이트웨이를 지나는 동안에만 보장됩니다. 예외는 Expedited Forwarding DiffServ 클래스 로, 이를 정의하면 자동으로 최고 우선순위 LLQ 클래스 가 되어 QoS 안에서도, 네트워크에서도 DiffServ 표시에 따른 대우를 받습니다(지연 강제 없이 DiffServ로만 쓰려면 Maximal Delay를 99999로).

언제 무엇을 쓸지는 ISP에 달렸습니다. ISP가 DiffServ를 지원하거나 MPLS로 여러 Class of Service를 제공 한다면, LLQ로 지연만 묶지 말고 DiffServ 클래스로 트래픽을 표시 해 ISP에 원하는 등급을 알려야 합니다.

09 QoS 관리

Managing QoS

이 장은 QoS를 실제로 구성하고 운영하는 작업들 을 모았습니다. 전역 속성과 인터페이스 속성을 잡는 일, 정책과 규칙을 만들고 고치는 일, DiffServ·LLQ 클래스를 인터페이스에 거는 일, 그리고 로깅을 켜는 일까지가 한 흐름입니다. 모든 절차는 SmartConsole을 연 상태를 전제로 합니다.

QoS 전역 속성

전역 속성에는 규칙 파라미터의 기본값과 측정 단위 가 들어 있습니다. SmartConsole의 **Application Menu** > **Global properties** > **QoS** 에서 잡는데(이때 SmartDashboard는 닫아 두어야 합니다), 핵심은 세 가지입니다. **Maximum weight of rule**(규칙에 줄 수 있는 최대 가중치, 기본 1000), **Default weight of rule**(새 규칙·Default에 기본으로 주는 가중치), 그리고 **Unit of measure**(전송률 단위, 예: Bps) 입니다. **Set Default** 로 기본값을 저장합니다.

인터페이스 QoS 속성

QoS가 트래픽을 통제하려면 먼저 게이트웨이와 그 인터페이스를 네트워크 객체로 정의 해야 합니다. 그다음 인터페이스 속성 창의 **QoS 탭** 에서 Inbound·Outbound 활성 전송률을 정하고 DiffServ·LLQ 클래스를 지정 합니다. 이 QoS 탭은 게이트웨이의 General Properties에서 QoS가 선택된 경우에만 활성화됩니다.

설정은 SmartConsole에서 게이트웨이 객체를 열어 **Network Management** 로 간 뒤 (인터페이스 목록이 없으면 **Get Interface**), 대상 인터페이스를 더블클릭해 **QoS 탭** 에서 합니다. DiffServ·LLQ 클래스는 여기서 Add/Edit/Remove 합니다.

여기엔 실무에서 꼭 알아야 할 원칙 이 있습니다. WAN 쪽(또는 느린 망 쪽) 인터페이스를 active로 두는 것이 보통이고, 인터페이스가 둘뿐이면 WAN 연결 인터페이스에만 QoS를 켭니다. 그리고 **무엇보다** 정한 전송률(Rate)이 인터페이스의 실제 물리 용량과 맞아야 합니다 — QoS는 이를 자동으로 확인해 주지 않습니다. 정한 값이 물리 용량보다 작으면 그만큼만 쓰고 (남는 건 안 씀), 크면 QoS가 트래픽을 제대로 통제하지 못 합니다.

QoS 정책 다루기

QoS 정책은 Rule Base에 순서대로 놓인 규칙들 입니다. 우리가 만든 규칙과 자동 생성된 Default Rule로 이뤄지고, Default는 수정만 되고 삭제는 안 되며 항상 맨 마지막 에 있습니다. Rule Base는 패킷을 어떻게 다룰지(출발지·목적지·서비스·시간·로깅)를 정하며, 만든 정책은 관련 게이트웨이와 인터페이스에 설치 해야 효력이 생깁니다.

새 정책은 **File** > **Manage Policies and Layers** > **New** 에서 만듭니다. 이름 규칙은 튜토리얼에서 본 것과 같고(예약어·공백·선두 숫자·특수문자· `.pf` / `.W` 금지), **QoS** 를 선택한 뒤 Express/Recommended를 고르면 SmartDashboard가 자동으로 열립니다. 기존 정책은 **Security Policies** > **Manage Policies** 에서 QoS 정책을 더블클릭해 엽니다.

규칙 만들고 고치기

규칙은 SmartDashboard에서 다룹니다. Default Rule이 항상 맨 아래여야 하므로, 새 규칙은 맨 마지막 규칙 뒤를 빼고 어디든 넣을 수 있습니다. QoS 탭에서 원하는 위치를 잡고 Rule 메뉴·툴바·우클릭으로 규칙을 추가한 뒤 이름을 정하면, 전역 속성의 기본값으로 규칙이 생성 됩니다. 추가 위치는 메뉴에서 맨 아래(Bottom)·맨 위(Top)·현재 규칙 아래(Below)·위(Above)·서브 규칙(Add Sub-Rule) 중에 고릅니다.

규칙을 우클릭하면 규칙 추가·삭제·복사·잘라내기·붙여넣기, Class of Service 추가, 숨기기(Hide)·비활성화(Disable)·이름 변경 같은 명령을 쓸 수 있습니다. **Hide** 는 화면에서만 감추고 설치 때는 포함 되며, **Disable** 은 Rule Base에 보이되 집행되지 않 습니다.

> 권 장
>
> 대역폭이 빠듯한 환경에서 새 규칙을 더할 때는, 기본 가중치를 10으로 두세요. 10보다 작게 바꾸면 그 규칙이 대역폭을 완전히 못 받 게 될 수 있습니다.

규칙의 각 필드 수정

규칙의 필드는 원하는 만큼 몇 번이고 고칠 수 있습니다. **Source·Destination·Service** 는 해당 열을 우클릭해 **Add** 로 객체를 더하고(원하는 만큼, 단 Service의 URI for QoS 리소스는 규칙당 하나만), Edit·Delete·Cut·Copy·Paste로 다듬습니다. 마지막 객체를 지우면 Any로 대체 되며, **Where Used** 로 그 객체가 어디 쓰이는지 볼 수 있습니다. Source에는 **Add Users Access** 로 사용자 그룹을 더하고 위치 제한(No restriction / Restrict to)도 걸 수 있습니다.

리소스를 단 서비스를 더할 때는 **Add with Resources** 를 쓰며, QoS에는 URI for QoS 타입 리소스만 됩니다. URI를 쓸 땐 `http://` 같은 프로토콜 접두어를 빼 고 적습니다(전체 URI: `www.my-site.com/pic/qos.gif` , 상대 URI: `/pic/qos.gif`). QoS가 지원하는 정규표현식은 `a*b` 형태이며, 자세한 문법은 정규표현식 부록을 봅니다.

Action 은 Action 열을 우클릭해 **Edit Properties** 로 엽니다. Simple이면 암호화 트래픽 전용 여부·Rule Weight·Rule Limit·Rule Guarantee 를, Advanced면 거기에 Per connection limit, Guarantee Allocation(Per rule/Per connection), Number of guaranteed connections, Accept additional connections 가 더해집니다(Express에서는 Advanced 불가).

> 중 요
>
> Rule Weight를 0(또는 보장도 0)으로 두면 그 규칙이 대역폭을 완전히 잃 을 수 있습니다. 기본값 10 같은 비율을 유지하세요. 그리고 Rule Guarantee는 Rule Limit보다 클 수 없 고, Number of guaranteed connections × Per connection guarantee도 Rule Limit를 넘을 수 없 습니다.

> 참 고
>
> 가중치·보장을 쓰면 WFQ가 남는 대역폭을 backlog 규칙들에 나눠 줘 한 톨도 안 버립 니다. 다만 Rule Limit를 걸면 그 위로는 남는 대역폭을 쓰지 않 습니다.

이 밖에 **Track**(로깅: None·Log·Account), **Install On**(규칙을 집행할 인터페이스 — QoS가 General Properties에서 켜져 있고 인터페이스가 QoS 탭에 정의되어야 함), **Time**(집행

시간대), **Comment** 도 같은 방식(우클릭 → Add/Edit)으로 다룹니다. Action을 기본값으로 되돌리려면 **Reset to Default** 를 씁니다.

서브 규칙 정의와 보기

서브 규칙은 규칙 안에서 대역폭을 더 잘게 나누는 규칙입니다. 규칙 이름 열을 우클릭해 **Add Sub-Rule** 을 고르면, 기본값을 가진 새 서브 규칙과 서브 Default가 함께 자동 생성 됩니다. 수정·추가 방법은 일반 규칙과 같습니다. 서브 규칙은 QoS Rule Tree에서 부모 규칙을 펼쳐 보며, 서브 규칙 하나를 클릭하면 Rule Base에 그 묶음 전체가 나타납니다.

DiffServ 다루기

DiffServ 규칙도 일반 규칙처럼 QoS Class와 가중치를 갖고, 그 가중치는 규칙이 설치된 인터페이스에서만 적용 됩니다(개념은 고급 정책 관리 참고).

DiffServ Class of Service 는 SmartDashboard의 **Manage** > **QoS** > **QoS Classes** 에서 **New** > **DiffServ Class of Service** 로 만들며, 이름·코멘트·색·타입(미리 정의/사용자 정의)을 정합니다(DiffServ code는 읽기 전용 비트맵 표시). 여러 클래스를 묶는 **Class of Service Group** 도 같은 메뉴에서 만듭니다.

인터페이스에 DiffServ를 걸려면, 게이트웨이 객체의 **Network Management** 에서 인터페이스를 열어 **QoS 탭** > **Add** > **DiffServ Classes** > **Others** 로 클래스를 더하고, Inbound/Outbound를 활성화해 Rate를 잡은 뒤 Guaranteed bandwidth(우선 표시할 보장 대역폭)와 Bandwidth Limit(이 클래스 최대 대역폭) 를 (최소 한 방향) 설정합니다. Rule Base에는 규칙 이름의 **Add Class of Service** > **Above/Below** 로 클래스 헤더를 더하는데, 첫 클래스를 정의하면 그 바로 아래 Best_Effort 헤더 가 나타납니다.

Expedited Forwarding 클래스 속성은 인터페이스 QoS 탭의 Add/Edit에서 **DiffServ Classes** > **Expedited Forwarding** 으로 잡으며, Constant Bit Rate와 Maximal Delay 를 (최소 한 방향) 정합니다(지연을 넘긴 패킷은 드롭). 일반 **DiffServ 클래스** 속성은 **Others** 로 잡으며 Guaranteed bandwidth와 Bandwidth Limit 를 정합니다(예: 용량 256MB·제한 192MB면 192MB 초과분은 표시 안 됨).

LLQ 다루기

LLQ는 음성·영상 같은 지연 민감 애플리케이션을 위한 특수 클래스 입니다(개념은 고급 정책 관리 참고).

Low Latency Class 는 **Manage** > **QoS** > **QoS Classes** > **New** > **Low Latency Class of Service** 로 만들어 이름·코멘트·색·타입을 정합니다. 인터페이스에 걸려면 SmartDashboard를 닫은 상태 에서 SmartConsole의 게이트웨이 객체 → **Network Management** → 인터페이스 → **QoS 탭** 으로 가, Inbound/Outbound 활성화 후 **Add** > **Low Latency Classes** 로 클래스를 고르고 Constant Bit Rate와 Maximal Delay 를 (최소 한 방향) 정합니다(지연 초과 패킷은 드롭). Expedited Forwarding을 DiffServ처럼만 쓰려면 Maximal Delay를 99999로 둡니다.

게이트웨이 상태 보기

QoS Security Gateway 상태는 SmartConsole의 **Gateways & Servers** 에서 게이트웨이를 클릭하면 아래쪽 Summary 탭 에 나타납니다.

로그 수집 켜기

연결이 로그에 남으려면 두 조건 이 맞아야 합니다. 게이트웨이 속성의 Additional Logging에서 QoS 로깅 플래그가 켜져 있어야 하고(기본으로 켜짐), 그 연결에 매칭되는 규칙의 Track 필드가 Log 또는 Account 로 표시되어야 합니다. 규칙이 로깅되는지 확인하려면 SmartDashboard에서 규칙을 골라 Track 필드에 Log 또는 Account가 있는지 봅니다. 로그가 실제로 어떻게 보이는지는 Logs & Events에서 다룹니다.

10 Logs & Events

Logs & Events

규칙에 조건을 걸어 로그를 남기고, SmartConsole의 **Logs & Events** 로 그 로그를 보며 QoS 정책이 잘 듣는지 모니터링하는 것이 이 장의 주제입니다. 로깅을 켜는 방법은 QoS 관리에서 다뤘고, 여기서는 어떤 사건이 어떻게 기록되는지 를 봅니다.

무엇이 로그에 남나

QoS 로그는 크게 비-Accounting 로그(사건 기록) 와 Accounting 로그(통계 기록) 로 나뉩니다.

비-Accounting 로그에는 세 가지 대표 사건이 있습니다. **Connection Reject** 는 보장 연결 수를 넘겼거나 추가 연결을 안 받게 설정했을 때 연결이 거부된 기록 으로, 거부의 빌미가 된 규칙 이름이 남습니다(Recommended 전용). **Running Out of Packet Buffers** 는 인터페이스-방향의 패킷 버퍼가 소진된 기록 으로, 최대 12시간에 한 번 만 보고됩니다(Recommended 전용). **LLQ Packet Drop** 은 LLQ 연결에서 패킷이 떨어졌을 때 의 기록으로, 최대 5분에 한 번 보고되며 지연 만료로 떨어진 바이트 수·평균 지연·지터를 담습니다(Recommended 전용).

Accounting 로그는 통계입니다. **General Statistics** 는 인터페이스·방향별로 QoS를 지난 총 바이트 를(Recommended·Express 모두), **Drop Policy Statistics** 는 QoS 정책 때문에 연결에서 떨어진 총 바이트 를(Recommended 전용), **LLQ Statistics** 는 LLQ 연결의 통계 를(Recommended 전용) 담습니다.

연결이 로그에 남는 조건은 앞 장에서 본 그대로입니다 — 게이트웨이 속성에서 QoS 로깅 체크박스가 켜져(기본값) 있고, 매칭 규칙의 Track이 Log 또는 Account 여야 합니다.

로그 사건 예시

Connection Reject 로그 는 보장 연결 수를 넘겼고 "Accept additional non-guaranteed connections"가 꺼져 있을 때 남습니다. 로그에는 `rule_name: <class> <name>` 형식으로 규칙의 클래스와 이름이 함께 남습니다(예: `rule_name:Best_Effort→udp2`, action `reject`).

LLQ Drop 로그 는 LLQ 연결에서 패킷이 떨어질 때, 직전 로그 이후의 LLQ 정보를 계산해 남깁니다. 인터페이스-방향별 접두어(예: Server-In은 `s_in_`)로 떨어진 바이트(`llq_drops`), 평균 전송 지연(`llq_avg_xmit_delay`), 최대 지연(`llq_max_delay`), 지터(`llq_xmit_jitter`), 권장 지연(`llq_recommended_delay`) 이 남습니다. 이 중 `recommended_delay` 는 드롭을 최소로 하려면 LLQ 클래스 속성에 넣을 만한 기본 지연값이라 특히 쓸모 있습니다.

> 참 고
>
> 특정 인터페이스-방향에 로그가 안 보일 수 있습니다. QoS가 그 방향에 설치 안 됐거나, 그 방향에 패킷이 없었거나, 값이 0이라 의미가 없 는 경우입니다.

Pool Exceeded 로그 는 인터페이스-방향(ifdir) 풀의 지정 크기를 넘겼을 때 남으며, 풀 크기·인터페이스 이름·방향(outbound 등)이 기록됩니다(예: `Ifdir Memory Pool Exceeded Pool_size:8`).

Accounting 통계 로그 예시

Accounting 로그에는 항상 `segment_time` (정보를 모은 시점) 이 Information 열에 들어가고, 의미 있는 데이터만 한 로그 레코드에 모아 보여 줍니다.

General Statistics 는 방향별 전송 바이트입니다(예: `s_in_bytes:5768`, `s_out_bytes:154294`). **Drop Policy Statistics** 는 drop policy로 연결에서 떨어진 바이트로(예: `s_out_total_drops`, 그중 할당 초과로 떨어진 `s_out_exceed_drops`), 이 drop policy는 패킷 버퍼를 관리하는 WFRED가 맡습니다. **LLQ Statistics** 는 항목이 LLQ Drop 로그와 같지만, 직전 로그가 아니라 연결 시작부터 누적해 만듭니다.

11 자주 묻는 질문(FAQ)

FAQ

원문 FAQ에서 실무에서 가장 자주 부딪히는 질문 을 주제별로 추려 풀어 둡니다. 개념의 자세한 설명은 본문 해당 장으로 연결해 둡니다.

QoS 기본

Express와 Recommended 중 무엇을? — 정교한 기능·고급 QoS가 필요하면 Recommended, 기본 QoS만 필요하면 Express 입니다. Express는 성능이 더 좋고 CPU·메모리를 덜 씁 니다. Express→Recommended 전환은 되지만 반대는 안 되므로, 확신이 없으면 Express로 시작하세요(자세히는 QoS 소개).

규칙에 줄 수 있는 최대 가중치는? — 가중치는 상대값이라, 유일한 한계는 전역 속성의 Maximum weight of rule(기본 1000, 임의로 변경 가능) 뿐입니다. 다만 초기 가중치를 크게 잡을지 작게 잡을지가 중요 합니다. HTTP·FTP가 각 500인 정책에 SMTP(100)를 더하면 HTTP는 500/1100으로 거의 안 줄 지만, 각 2인 정책에 SMTP(100)를 더하면 2/104로 확 줄어듭니다. 즉 초기 가중치가 크면 규칙을 더해도 영향이 작 습니다.

QoS를 외부·내부 인터페이스 중 어디에? — 둘 다 되지만 외부 인터페이스에만 두는 것이 강력히 권장 됩니다.

보장과 가중치의 차이는? — 둘 다 매칭 트래픽에 대역폭을 보장한다는 점은 같지만, 보장은 절대값(예: 20000bps), 가중치는 상대값(예: 100) 입니다. 그리고 보장이 가중치보다 먼저 대역폭을 가져갑니다. 예컨대 1.5MB 회선에 HTTP Guarantee 1Mb·FTP Weight 40·SMTP Weight 10이면, 먼저 HTTP에 1MB, 남은 0.5MB를 FTP 0.4·SMTP 0.1 로 나눕니다.

보장은 대역폭을 낭비하나? — 아닙니다. 애플리케이션은 필요한 만큼만 쓰고, 안 쓰는 대역폭은 즉시(패킷 단위로) 다른 연결에 상대 우선순위대로 재분배 됩니다. QoS에는 "빌려준 대역폭"이라는 개념이 없습니다.

TCP 재전송은 어떻게? — 재전송을 감지하면 그 데이터가 이미 QoS 큐에 있는지 확인해, 있으면 패킷을 떨굽 니다(RDED). 이 덕에 WAN 링크의 최대 40%를 잡아먹는 재전송을 없애고 메모리도 아낍니다.

다른 Check Point 제품과의 관계

QoS는 Multi-Domain Security Management와 정교하게 통합 됩니다 — VPN 터널 안 트래픽의 정확한 분류, NAT 트래픽 분류, 네트워크 객체·토폴로지 공유 등. SmartView Monitor는 QoS의 일부가 아니라 함께 번들되는 별도 제품 이고, QoS는 모든 ClusterXL 구성 (Load Sharing 포함) 과 NAT 트래픽 을 완전히 지원합니다. 관리 가능한 QoS 게이트웨이 최대 수는 일반 게이트웨이와 동일 하며, QoS 게이트웨이를 관리하려면 Management Server에도 QoS를 설치 해야 합니다.

정책 만들기

LLQ는 언제? — VoIP·화상회의 등 멀티미디어 처럼 최소 보장 대역폭이 필요하고 낮은 지연·지터가 중요 한 경우입니다(고급 정책 관리).

Rule Base는 first match인가? — 예. 연결에 처음 맞는 규칙만 적용 됩니다. 그래서 CEO 트래픽 규칙과 HTTP 규칙이 있을 때, CEO를 보장하려면 CEO 규칙을 HTTP 규칙보다 위에 둬야 합니다(아래 두면 CEO의 웹서핑이 HTTP 제한에 걸림).

여러 게이트웨이의 Rule Base 정리법은? — 대역폭이 같고 정책도 같으면 Install On을 All 로, 대역폭이 다른데 정책이 같으면 (절대값 아닌) 가중치를 써서 하나의 정책 을, 대역폭도 정책도 다르면 게이트웨이별 서브 규칙 을 씁니다.

서브 규칙은 언제? — 객체 사이에 위계가 있을 때 — 예컨대 R&D·Marketing·운영 같은 조직 구조대로 대역폭을 나눌 때입니다. **대역폭을 많이 먹는 앱은?** — CLI에서 `rtmtopsvc` 를 실행해 봅니다(CLI 참조).

용량 산정과 튜닝

메모리 요구량은? — 연결 수에 비례합니다. 5,000 연결에 약 32.5MB, 50,000에 약 91MB, 100,000에 약 156MB 가 게이트웨이에 추가로 필요합니다(평균 연결당 ~1300바이트, 기본 연결 테이블 25,000). **하드웨어는?** — 성능의 주 요인은 CPU 입니다. 메모리는 값이 싸고 발자국이 작아 보통 병목이 아닙니다.

성능 튜닝 팁 — 최신 QoS 버전으로 업그레이드, 대개 외부 인터페이스에만 설치, inbound 제한을 안 쓰면 outbound 방향에만 설치, 자주 쓰는 규칙을 Rule Base 위로, per-connection 제한·보장을 per-rule로 전환 하는 것입니다. **최대 지원 대역폭은?** 10Gbps 입니다.

설치·버전·실무 시나리오

메일/ERP 서버 성능 보장은? — 해당 트래픽(출발지·목적지 또는 SMTP·POP3, SAP·ORACLE 등 프로토콜)에 매칭되는 규칙을 하나 만들고 가중치나 보장 을 줍니다. **VoIP 품질 보장은?** — LLQ에 per-connection 보장 을 써서 통화마다 대역폭을 보장하고, 대역폭이 부족하면 추가 통화를 안 받는 입장 정책 을 둡니다(옛 전화의 통화 중 신호와 같은 개념).

QoS로 DoS 공격을 막나? — QoS는 DoS 전용 도구는 아니지만, DoS에 흔히 쓰이는 앱(ICMP·수상한 URL)을 제한 하고 중요 트래픽(ERP·메일·VoIP)에 보장 을 줘 탐지·모니터링·예방에 쓸 수 있습니다. **차단보다 대역폭 제한이 나은 이유는?** — 완전 차단은 사용자가 우회로를 찾게 만들 어 법적 문제까지 부를 수 있습니다. 대역폭을 제한하거나 시간대별로 조절 하면 업무를 해치지 않으면서 통제할 수 있습니다.

일반적인 문제

SmartView Monitor에서 보장/제한이 깨져 보입니다 — 낮은 제한(예: 1000Bps)을 잦은 주기(2초)로 보면 QoS가 패킷을 쪼개지 않아 그렇게 보일 수 있습니다. 샘플링 주기를 8초로 낮추면 제한이 지켜지는 것을 볼 수 있습니다.

정의 대역폭이 실제 물리 대역폭보다 작으면/크면? — 작으면 정의한 만큼만 쓰고 나머지는 안 씁 니다. 크면 물리 한계 이상을 보내려다 다음 홉에서 무작위 드롭 이 생겨 중요한 패킷을 잃을 수 있습니다. 그래서 QoS 관리에서 강조했듯 정의 전송률을 실제 용량에 맞추 는 것이 중요합니다.

12 명령줄 참조

Command Line Reference

QoS는 SmartConsole·SmartDashboard로 거의 다 다루지만, 게이트웨이에서 직접 블레이드를 켜고 끄거나 정책을 가져오고 상태를 볼 때 는 CLI가 필요합니다. 이 장은 QoS의 핵심 명령 — `etmstart`·`etmstop`·`fgate` — 를 정리합니다. QoS의 내부 데몬 이름이 **fgd50**(FloodGate-1의 흔적)이라 명령에도 `fg` 가 자주 보입니다.

> 중 요
>
> Scalable Platforms(Maestro·Chassis)에서는 Expert 모드에서 해당 Security Group 위에서 명령을 실행해야 합니다.

구문 표기 약속

CLI 구문에는 몇 가지 기호 약속 이 있습니다. 중괄호 `{ }` 는 `|` 로 구분된 선택지 중 하나, 꺾쇠 `< >` 는 사용자가 채워야 하는 변수, 대괄호 `[ ]` 는 생략 가능한 선택 항목 을 뜻합니다. 들여쓰기(TAB)는 중첩된 하위 명령을 나타냅니다.

etmstart — QoS 시작

`etmstart` 는 게이트웨이에서 QoS 블레이드를 시작 합니다. QoS 데몬 fgd50을 띄우고, `$FWDIR/conf/masters` 에 설정된 Management Server에서 QoS 정책을 가져 옵니다.

```
[Expert@MyGW:0]# etmstart
QoS: Starting fgd50
QoS: Fetching QoS Policy from masters
Fetching QoS Software Blade Policy:
Received Policy. Downloading...
eth0(inbound), eth0(outbound).
Download OK.
Done.
QoS started
```

etmstop — QoS 정지

`etmstop` 은 반대로 QoS 블레이드를 멈춥 니다. 데몬 fgd50을 죽이고 QoS 정책을 내려(unload) 놓습니다. 데몬을 수동으로 재시작하려면 `etmstop` 뒤에 `etmstart` 를 실행합니다.

```
[Expert@CXL1_192.168.3.52:0]# etmstop
Unloading QoS Policy:
...
QoS policy unloaded successfully.
QoS stopped
```

fgate — 정책 설치·상태·디버그

`fgate` 는 실행 위치에 따라 쓰임이 조금 다릅니다.

Management Server에서 는 관리하는 게이트웨이에 QoS 정책을 설치·제거하고 상태를 봅니다. `fgate load <정책명>.F <GW1> <GW2> …` 는 정책을 검증한 뒤 컴파일해 지정 게이트웨이에 설치 하고(정책명은 최대 32자, 게이트웨이는 메인 IP나 객체 이름으로 지정), `fgate unload <GW…>` 는 제거, `fgate stat <GW…>` 는 상태, `fgate ver` 는 버전을 보여 줍니다.

```
[Expert@MGMT:0]# fgate load MyPolicy.F 192.168.3.52
QoS rules verified OK!
Downloading QoS Policy: MyPolicy.F...
MyGW: QoS policy installed succesfully.
Done.
```

Security Gateway에서 는 여기에 정책 가져오기와 디버그 제어 가 더해집니다. 자주 쓰는 것만 추리면 — `fgate fetch -f` 는 masters 파일의 모든 Management Server에서 정책을 가져와 설치(특정 서버는 `fgate fetch <서버>`), `fgate load` 는 로컬 정책 설치(실패하면 `etmstop` → `etmstart`), `fgate unload` 는 제거, `fgate stat [-h]` 는 상태, `fgate ver [-k]` 는 버전(`-k` 면 커널 버전도)입니다.

이 밖에 세밀한 제어용 옵션이 있습니다. `fgate ctl <모듈> {on|off}` 는 QoS 모듈 제어 (R82에서 모듈은 `etmreg` 뿐), `fgate debug {on|off}` 는 fgd50 데몬 디버그 로 추가 정보를 `$FGDIR/log/fgd.elg` 에 남기고, `fgate log {on|off|stat}` 는 커널의 QoS 로깅 상태 를 제어합니다(정책 재설치 없이 로깅을 꺼 리소스 절약 가능). `fgate kill [-t <신호>] <프로세스>` 는 QoS 프로세스에 신호를 보냅 니다(R82의 프로세스는 fgd50 뿐, 신호를 안 주면 기본 15(SIGTERM)).

```
[Expert@MyGW]# fgate fetch -f
Fetching QoS Software Blade Policy:
Received Policy. Downloading...
Download OK.
Done.
```

참 고

fgate stat 는 옛 버전 호환용으로 남아 있는 구식 명령 입니다. 클러스터에서는 모든 멤버를 똑같이 설정 해야 하며, 자세한 제어·디버그는 sk41585("How to control and debug QoS / FloodGate-1")를 참고하세요.

13 커널 파라미터 다루기

Working with Kernel Parameters

QoS도 다른 Software Blade처럼 커널 파라미터로 세밀한 동작을 조정 할 수 있습니다. 다만 이 주제는 QoS 전용이 아니라 모든 Security Gateway에 공통 이라, QoS 가이드에서 따로 다루지 않고 게이트웨이 가이드로 넘깁니다.

자세한 절차는 **R82 Quantum Security Gateway Guide** 의 "Working with Kernel Parameters" 장 을 참고하세요. 커널 파라미터를 보고·바꾸고·재부팅 후에도 유지되게 하는 방법이 거기에 정리되어 있습니다.

이어지는 Kernel Debug도 같은 게이트웨이 가이드를 가리킵니다.

14 커널 디버그

Kernel Debug

QoS가 커널 수준에서 어떻게 동작하는지 깊이 파고들어야 할 때는 커널 디버그 를 씁니다. 이 역시 QoS 전용이 아니라 Security Gateway 공통 주제 라, QoS 가이드에서는 안내만 하고 게이트웨이 가이드로 넘깁니다.

자세한 절차는 **R82 Quantum Security Gateway Guide** 의 "Kernel Debug on Security Gateway" 장 을 참고하세요.

참고로 QoS 데몬(fgd50) 수준의 디버그는 CLI 참조에서 본 `fgate debug on` 으로 켜며, 그 결과는 `$FGDIR/log/fgd.elg` 에 남습니다. 더 깊은 절차·해석은 sk41585를 함께 봅니다.

15 부록: 정규표현식

Appendix: Regular Expressions

QoS 규칙에서 URI for QoS 리소스로 HTTP 트래픽을 URL 패턴으로 매칭 할 때 정규표현식을 씁니다(QoS 관리의 서비스·리소스 참고). 이 부록은 Check Point가 구현한 정규표현식 문법과, 덤으로 R77 정책용 QoS 가속을 끄고 켜는 절차 를 정리합니다.

정규표현식 문법

Check Point는 표준 정규표현식 메타문자를 거의 그대로 따릅니다. `\` 는 메타문자·비출력 문자·문자 종류를 escape 하고, `[ ]` 는 문자 클래스 정의, `( )` 는 하위 패턴(묶음에 메타문자 적용), `.` 은 임의의 한 문자 입니다. 수량을 정하는 기호로는 `?` (0~1회), `*` (0회 이상), `+` (1회 이상) 가 있고, `{n}` 은 정확히 n회, `{n,m}` 은 n~m회, `{n,}` 은 n회 이상 을 뜻하는 중괄호 수량자입니다. 그 밖에 `|` 는 택일, `^` 는 버퍼(보통 단어) 시작, `$` 는 끝에 고정, `-` 는 문자 클래스 안의 범위 입니다.

> 참 고
>
> 서비스·리소스에서 언급했듯, QoS가 흔히 쓰는 형태는 a*b(a·b는 문자열, *는 와일드카드)입니다.

비출력 문자

출력되지 않는 문자를 패턴에 쓰려면 예약 문자를 escape 합니다. 대표적으로 `\n` 은 newline(0A), `\r` 은 carriage return(0D), `\t` 는 tab(09), `\f` 는 formfeed(0C), `\e` 는 escape(1B), `\a` 는 alarm/BEL(07) 입니다. 코드로 직접 지정할 때는 `\ddd` 는 8진 코드, `\xhh` 는 16진 코드, `\cX` 는 "control-X" 를 나타냅니다.

문자 종류

특정 종류의 문자 를 가리킬 때도 escape를 씁니다. `\d` 는 10진 숫자 `[0-9]`, `\D` 는 숫자가 아닌 문자, `\s` 는 공백 문자, `\S` 는 공백이 아닌 문자, `\w` 는 단어 문자(밑줄·영숫자), `\W` 는 단어가 아닌 문자 입니다.

QoS 가속 끄기·켜기

QoS 소개에서 봤듯, R77 이전용으로 만든 QoS 정책에서 옛 기능을 쓰려면 QoS 가속을 꺼야 합니다.

끄는 절차는 이렇습니다. 게이트웨이에서 `cpconfig` 로 SecureXL·CoreXL을 끄고 재부팅 한 뒤, 다음 명령을 실행합니다.

```
cpprod_util CPPROD_SetValue FG1 FgWithAcceleration 1 0 1
```

다시 켤 때는 순서가 반대입니다. 먼저 아래 명령을 실행한 뒤, `cpconfig` 로 SecureXL/CoreXL을 켜고 재부팅 합니다.

```
cpprod_util CPPROD_SetValue FG1 FgWithAcceleration 1 1 1
```