# Class-Based Policy Provisioning:

# Introducing Class-Based Policy Language (CPL)

**August 2008**

# Class-Based Policy Provisioning

- Introduction
- Class-Based Policy Provisioning
- Class-Based Policy Language (CPL)
- Integrated Traffic Classification
- Configuring with CPL:  Examples
- Monitoring and Statistics
- Roadmap
- Q&A

# Many Features Act on Traffic

- Many features need to understand network traffic
  - Quality of Service
  - Security
  - Broadband
  - NetFlow
  - Routing
  - … and many others
- Issue: Each feature might take a unique approach
  - ➢ Different configuration command syntax
  - ➢ Unnecessary complexity for customers

# The Opportunity

- Simplify

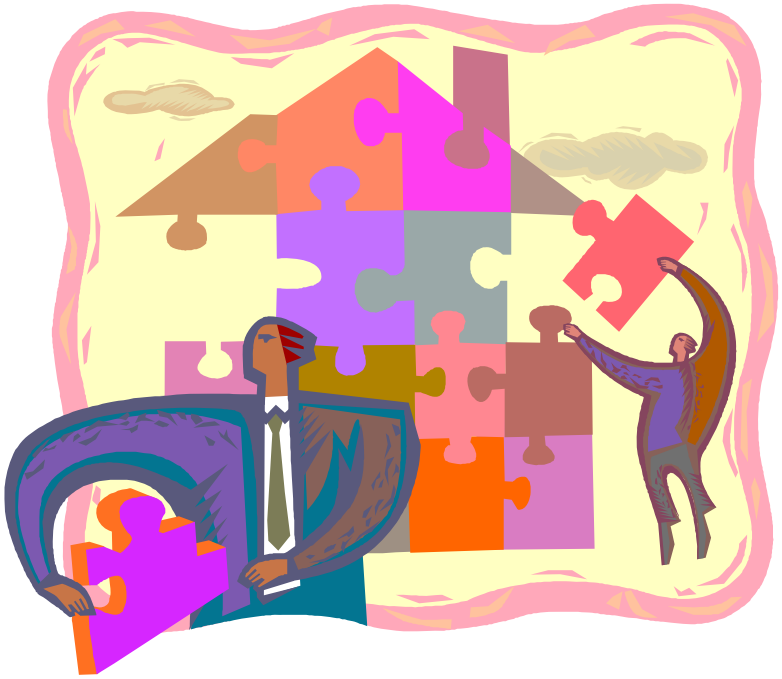
  Simplify feature provisioning

- Unify

  Unify provisioning and behavior across platforms

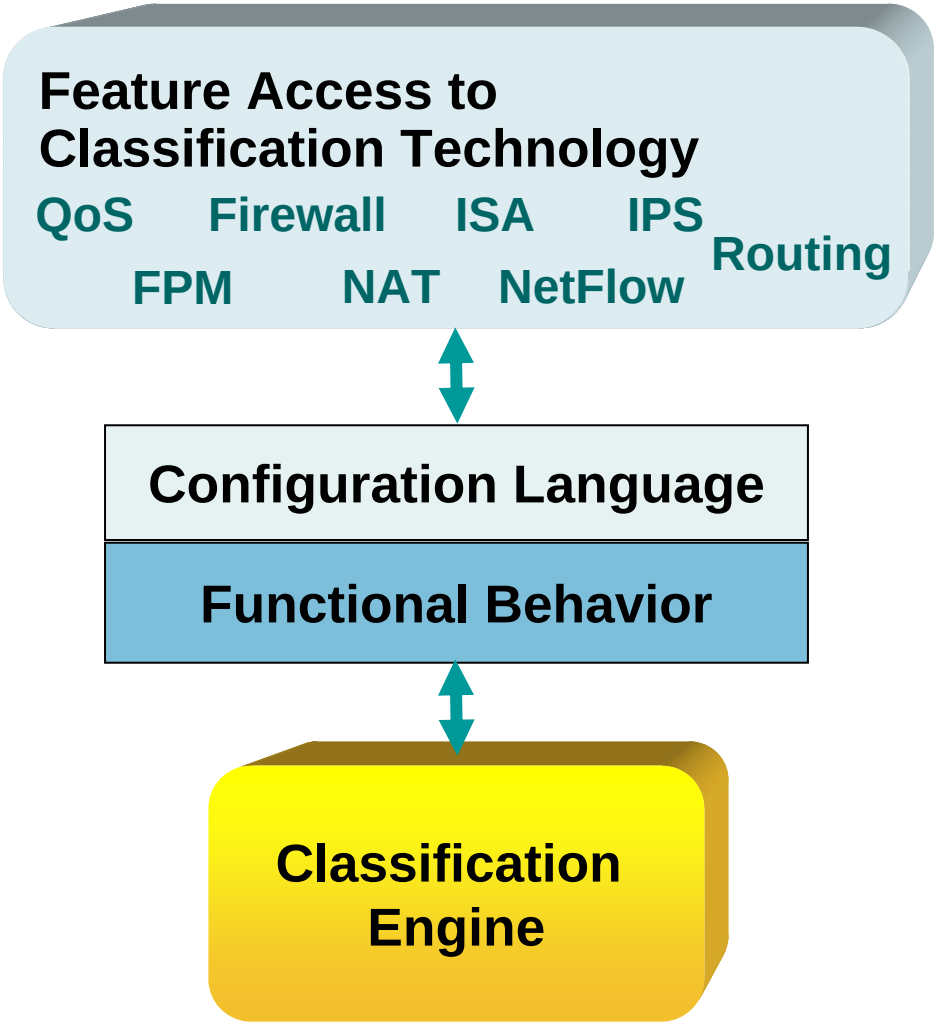
- Integrate

  Make it easy to add new function and new platforms



# Uniform Provisioning for Traffic Classification and Policy Actions

- Uniform Provisioning
  - Across Features
  - Across Platforms
- Unified Configuration Language
- Integrated Classification Definitions
- Greater Efficiency



# CLASS-BASED POLICY PROVISIONING

# Class-Based Policy Provisioning

A uniform, three step approach

- Classification
  - Identify traffic of interest
  - Specify match criteria that define a traffic class
- Policy
  - Specify actions to take on the traffic class
- Target
  - Apply the policy actions to a target
    - Typically an interface or subinterface

Class-Based Policy Provisioning

- *Classification*
  - *Key word: class-map*
- *Policy*
  - *Key word: policy-map*
- *Target*
  - *Key word: service-policy*

# Class-Based Policy: Terminology

- What is a *class* of traffic?
  - A class is any traffic stream of interest
    - Identify traffic streams by matching some criteria, such as
      - From a particular interface or port
      - Source or destination IP address
      - Protocol or application
- What is a *policy*?
  - A policy is any action applied to a class
    - Policies for Quality of Service, Security, Routing, Accounting, or Subscriber Service, such as
      - Assign higher or lower priority
      - Limit or drop traffic
      - Route on a different path

**Example**

**Certain apples (the *class*) are selected for special handling (the *policy*)**

# Class-Based Policy: Terminology

- What is a *target*?

  A target defines a traffic stream to which a policy is applied

- Typically identifies the location, source or destination of traffic
  - Physical interfaces
    - Serial, Ethernet, POS
  - Logical interfaces
    - Subinterface, ATM VC, Frame Relay VC, VLAN
  - Logical entities
    - Control Plane Traffic
    - A Routing Protocol

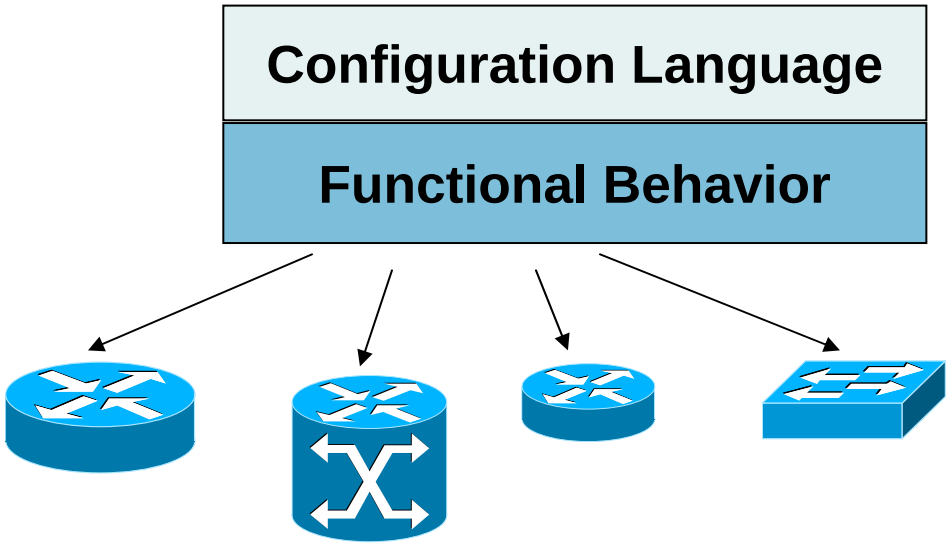
**Example**

**An orchard could be a source (target) to which a harvest policy is applied**

# Class-Based Policy Language (CPL)

- Unified method to specify classes, policies and targets
- Same framework for provisioning multiple features

  "Type" attributes for classes & policies




Benefit: Simpler provisioning

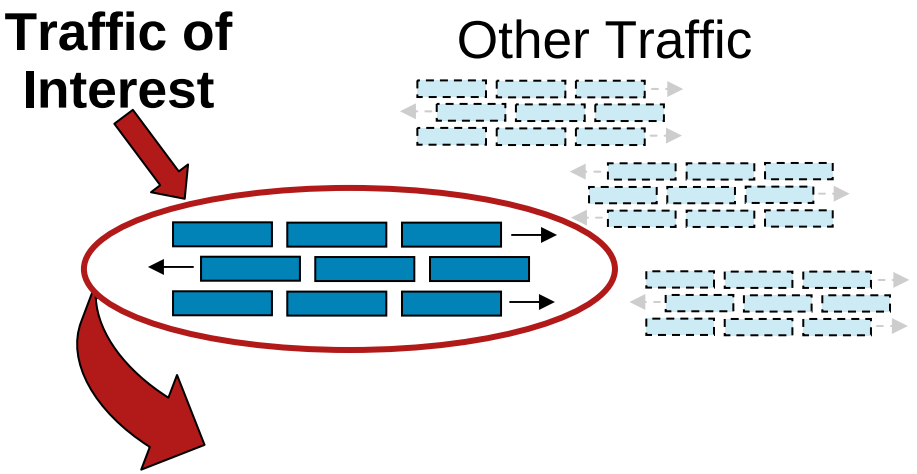
Across features

Across platforms

# Specify Traffic Once, Take Multiple Actions

- With CPL features can share a *class-map*
  - Set up the classification criteria once
  - Use the *class-map* in different feature policies
- Benefits
  - Simplified configuration – policies point to same classification
  - Assured consistency – actions applied to same traffic

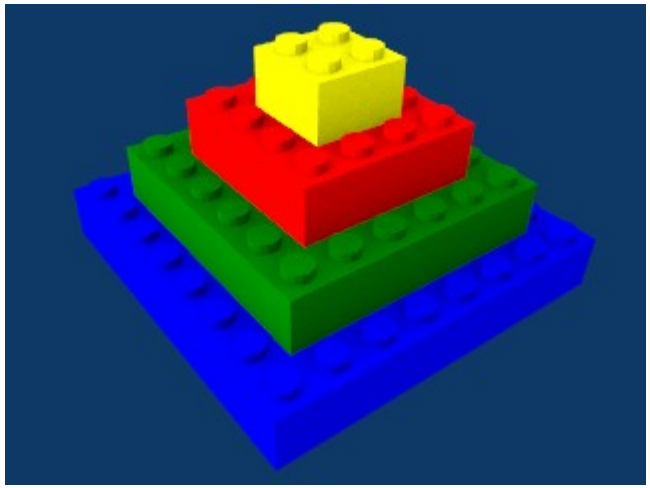
**Example:**




- Firewall policy permits selected traffic
- QoS policy assigns priority

# Class-Based Policy Framework: Benefits

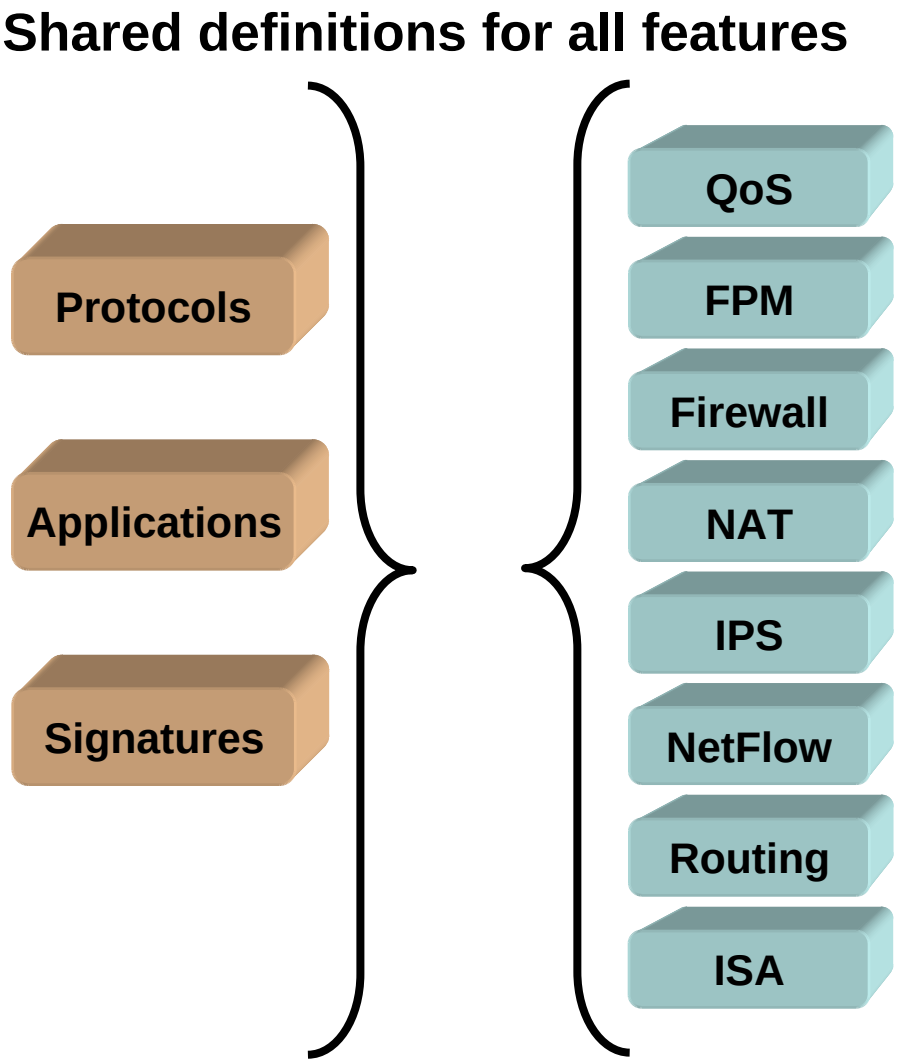
- Simpler for customers
  - Unified method for multiple features
- Faster time-to-market for new application recognition modules
  - Leverage by multiple features
- Easier to add new features
  - Leverage familiar provisioning method
    - Example (future): anomaly detection
- Quicker integration of new classification capabilities
  - Directly available to existing features
  - Enable new policy actions



# INTEGRATED TRAFFIC CLASSIFICATION

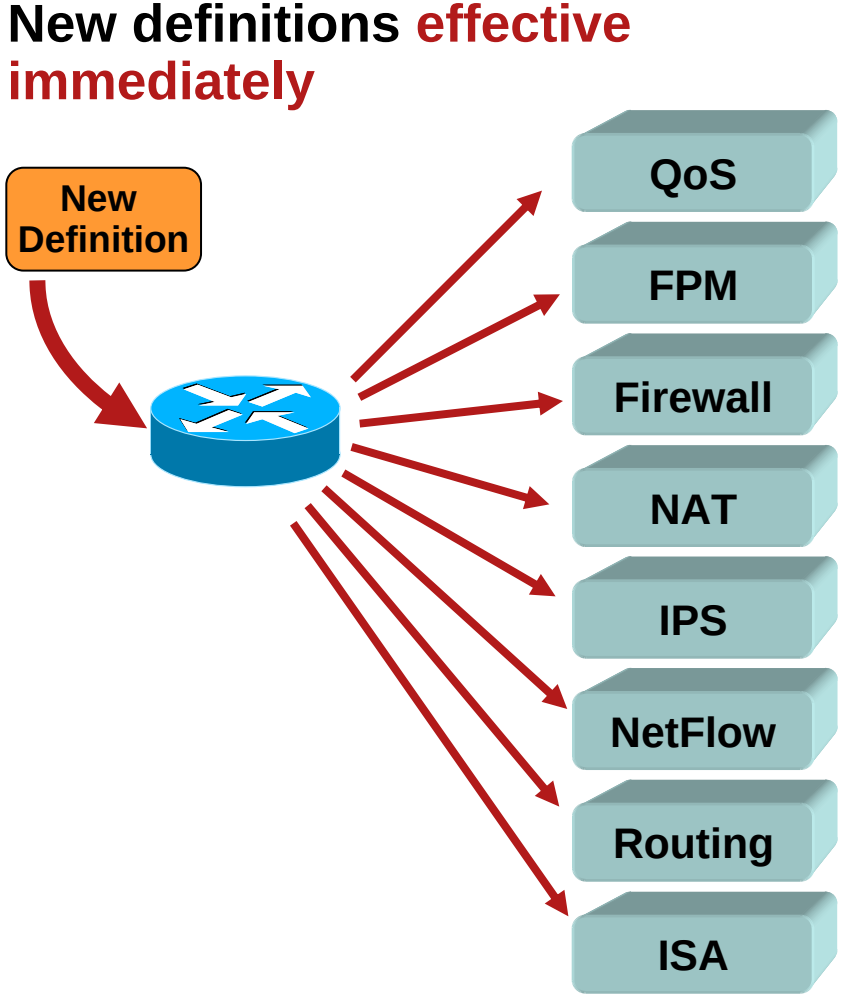
# Benefit: Integrated Classification Definitions

- Common definitions for protocol and application recognition
- Benefits:
  - Consistent classification results
  - New definitions available to all features



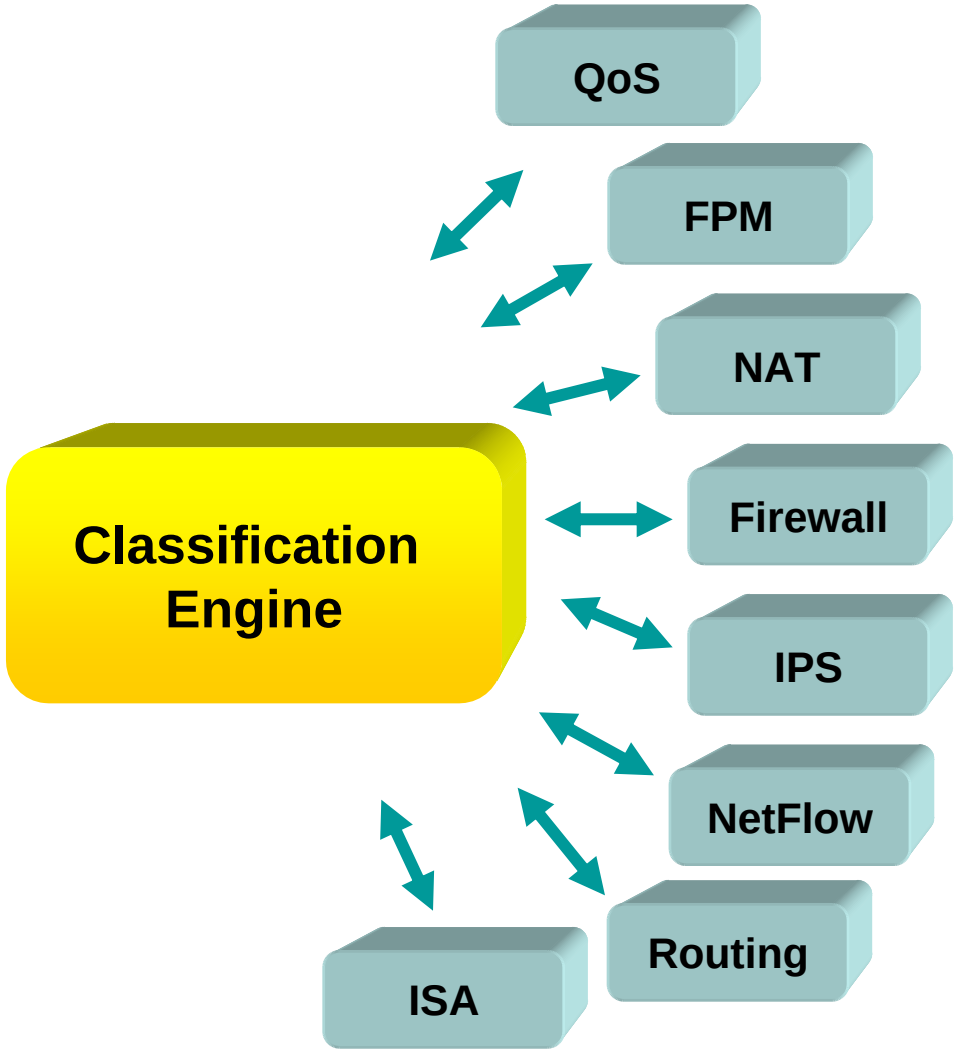
# Benefit: Dynamic Availability of New Definitions

- Dynamic addition of new definitions
  - Immediately available to all features
  - Live updates to in-service routers
    - Incorporate new definitions into live IOS images on the router



# Benefit: Greater Efficiency

- Performance improvement
  - Consolidated classification for multiple features
    - Not separate classification actions for each feature
- Benefits:
  - Lower CPU consumption
  - Greater throughput



# CPL CONFIGURATION EXAMPLES

# CPL Configuration Examples

- Quality of Service
- Flexible Packet Matching
- IOS Firewall
- Multi-Topology Routing
- IP SLAs
- NetFlow
- Summary Comparison:

  QoS, FPM, Firewall, and MTR

# CPL Configuration Example
# QoS Giving Priority to Interactive Traffic

**Classification**
Key word:
**class-map**

Configure *class-maps* that classify Citrix ICA traffic by ICA tag

```
class-map match-any Citrix-high-medium-low
        match protocol citrix ica-tag “0”
        match protocol citrix ica-tag “1”
        match protocol citrix ica-tag “2”
class-map Citrix-background
        match protocol citrix ica-tag “3”
```

**Policy**
Key word:
**policy-map**

Create a *policy-map* that allocates bandwidth for traffic matched by *class-maps*

```
policy-map Citrix-traffic
        class Citrix-high-medium-low
                bandwidth percent 20
        class Citrix-background
                bandwidth percent 5
                police cir 128000
                        conform-action transmit
                        exceed-action drop
```

**Target**
Key word:
**service-policy**

Assign the *policy-map* to a router interface with a *service-policy*

```
interface serial 0/0
service-policy output Citrix-traffic
```

# *CPL Configuration Example*
# FPM Used to Drop Slammer Worm

**Classification**
Key word:
**class-map**

Configure *class-maps* that classify Slammer worm

```
class-map type stack ip-udp
    match field ip protocol eq 17 next udp
class-map access-control slammer
    match class-map stack ip-udp
    match field udp dport eq 1434
    match start ip version offset 224 size 4 eq 0x04011010
```

**Policy**
Key word:
**policy-map**

Create a *policy-map* that drops traffic matched by the *class-map*

```
policy-map type access-control policy-slammer
    class slammer
        drop
```

**Target**
Key word:
**service-policy**

Assign the *policy-map* to a router interface with a *service-policy*

```
interface ethernet 1/0  service-policy type access-control input
policy-slammer
```

# CPL Configuration Example
# IOS Firewall Blocks Instant Messaging

**Classification**
Key word:
**class-map**

Configure *class-maps* to identify port-misuse and classify HTTP

```
class-map type inspect http port-misuse-class
      match port-misuse im
class-map type inspect http-traffic-1
      match protocol http
```

**Policy**
Key word:
**policy-map**

Create *policy-maps* to terminate IM connections but permit desired HTTP traffic

```
policy-map type inspect http myL7policy
        class port-misuse-class
          reset
policy-map type inspect firewall-policy
        class http-traffic-1
          inspect
```

**Target**
Key word:
**service-policy**

Assign the *policy-map* to a router interface with a *service-policy*

```
interface pos 0/0
service-policy type http myL7Policy
```

# *CPL Configuration Example*
# Multi-Topology Routing

## Video traffic routed separately from other traffic

**Classification**
Key word:
**class-map**

```
class-map match-any STANDARD_CLASS
 match ip dscp default
class-map match-any VIDEO_CLASS
 match ip dscp af43
```

**Policy**
Key word:
**policy-map**

```
policy-map type class-routing ipv4 unicast MTR_POLICY
 class STANDARD_CLASS
  select-topology STANDARD
 class VIDEO_CLASS
  select-topology VIDEO
```

**Target**
Key word:
**service-policy**

```
global-address-family ipv4
 topology STANDARD
 !
 topology VIDEO
 !
 service-policy type class-routing MTR_POLICY
```

Source: Configuration fragment, MTR demo 9/2005

# CPL Configuration Example
# IP SLAs Integrated with a QoS Policy

## Using IP SLAs to monitor a traffic class

**Classification**
Key word:
**class-map**

```
class-map match-any VOIP
      match ip dscp EF
class-map match-any biz
      match ip dscp AF41
class-map default
```

**Policy**
Key word:
**policy-map**

```
policy-map high-priority
      class VOIP
            measure gold-sla
            bandwidth 2000
```

**IP SLAs action within a policy-map**

**key word: measure**

```
ip sla auto-measure gold-sla
      measurement-type jitter
      dest-ip auto-discover
```

**IP SLAs configuration**

**Target**
Key word:
**service-policy**

```
interface serial 0/0
      service-policy output high-priority
```

# *CPL Configuration Example*
# NetFlow Input Filters

## NetFlow sampling actions in CPL

**Classification**
Key word:
**class-map**

```
class-map high_importance_class
    match access-group 101
```

**Defines traffic class**

```
flow-sampler-map high_sampling
    mode random one-out-of 1
```

**Defines a NetFlow sampler**

**Policy**
Key word:
**policy-map**

```
policy-map mypolicy
    class high_importance_class
    flow-sampler high_sampling
```

**Includes NetFlow sampling action in policy**

**Target**
Key word:
**service-policy**

```
interface POS1/0
    service-policy input mypolicy
interface ATM2/0
    service-policy input mypolicy
```

**Applies policy with Netflow sampling action to interfaces**

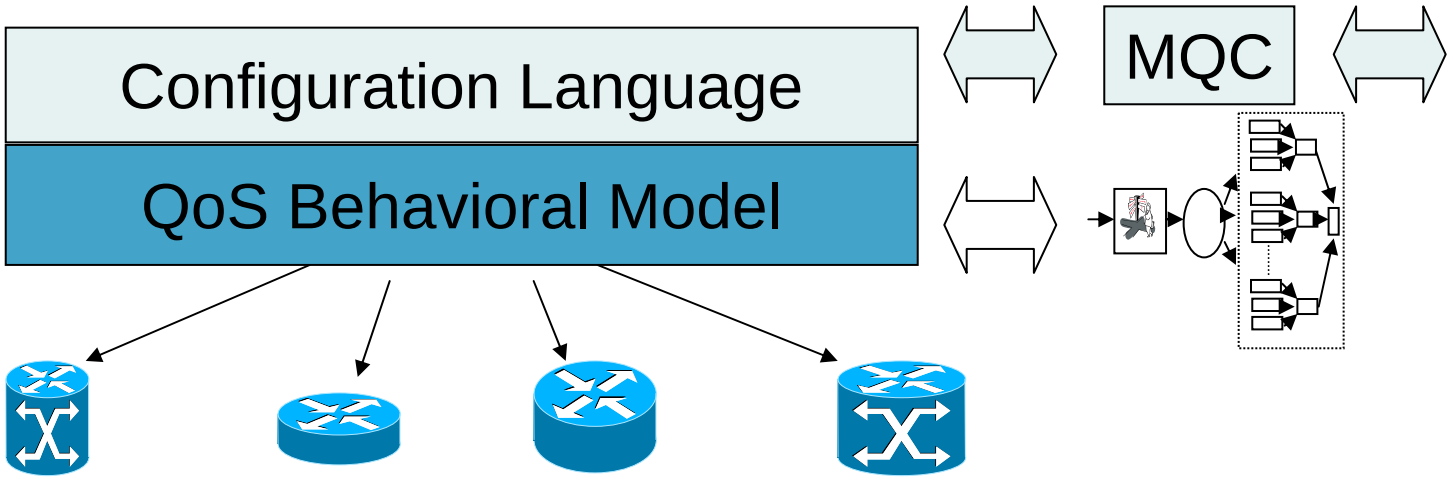
# CPL Configuration Comparisons: Classification, Policy, Target

| CPL Step<br>*key word* ***[type [subtype] ]*** | Quality of Service<br>**Prioritize Citrix Traffic** | Flexible Packet Matching<br>**Drop Slammer Worm** | IOS Firewall<br>**Block Instant Messaging** | Multi-Topology Routing<br>**Assign Routes by Class** |
|---|---|---|---|---|
| Classification<br>*class-map* ***[type [subtype] ]*** | class-map match-any Citrix-high-medium-low<br>match protocol citrix ica-tag “0”<br>match protocol citrix ica-tag “1”<br>match protocol citrix ica-tag “2”<br>class-map Citrix-background<br>match protocol citrix ica-tag “3” | class-map type stack ip-udp<br>match field ip protocol eq 17 next udp<br>class-map access-control slammer<br>match class-map stack ip-udp<br>match field udp dport eq 1434<br>match start ip version offset 224 size 4 eq 0x04011010 | class-map type inspect http port-misuse-class<br>match port-misuse im<br>class-map type inspect http-traffic-1<br>match protocol http | class-map VIDEO_CLASS<br>match {VIDEO DSCP value}<br>class-map VOICE<br>match {VOICE DSCP EF}<br>class-map DATA<br>match {DATA DSCP value} |
| Policy<br>*policy-map* ***[type [subtype] ]*** | policy-map Citrix-traffic<br>class Citrix-high-medium-low<br>bandwidth percent 20<br>class Citrix-background<br>bandwidth percent 5<br>police cir 128000<br>conform-action transmit<br>exceed-action drop | policy-map type access-control policy-slammer<br>class slammer<br>drop | policy-map type inspect http myL7policy<br>class port-misuse-class<br>reset<br>policy-map type inspect firewall-policy<br>class http-traffic-1<br>inspect | policy-map type class-routing MTR_ROUTE_POLICY<br>class VIDEO<br>select-topology RED<br>class VOICE<br>select-topology YELLOW<br>class DATA<br>select-topology GREEN |
| Target<br>*service-policy* ***[type [subtype] ]*** | interface serial 0/0<br>service-policy output Citrix-traffic | interface ethernet 1/0<br>service-policy type access-control input policy-slammer | interface pos 0/0<br>service-policy type http myL7Policy | global-address-family ipv4<br>service-policy type MTR_ROUTE_POLICY |

# MONITORING AND STATISICS: CLASS-BASED POLICY (CBP) MIB

# CBQoS MIB Shadows QoS Configuration

- Modular QoS Command Line Interface (MQC) is Cisco's configuration language for Quality of Service
  - Uniform interface for common QoS model across hardware platforms
- CBQoS MIB provides read access to configuration and statistical information for MQC
- Same structure as MQC




CBQoS MIB

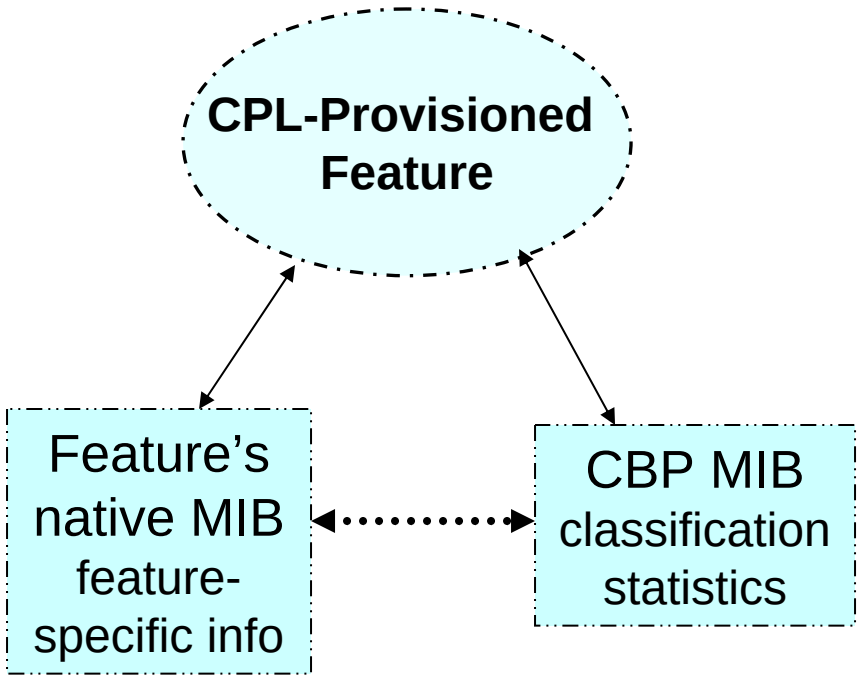
```
1 :  cbQosServicePolicy
2 :  cbQosInterfacePolicy
3 :  cbQosFrameRelayVCPolicy
4 :  cbQosATMPVCPolicy
5 :  cbQosObjects

6 :  cbQosPolicyMapCfg
7 :  cbQosClassMapCfg
8 :  cbQosMatchStmtCfg
9 :  cbQosQueueingCfg
10:  cbQosREDCfg
11:  cbQosREDClassCfg
12:  cbQosPoliceCfg
13:  cbQosTSCfg
14:  cbQosSetCfg

15:  cbQosClassMapStats
16:  cbQosMatchStmtStats
17:  cbQosPoliceStats
18:  cbQosQueueingStats
19:  cbQosTSStats
20:  cbQosREDClassStats
```

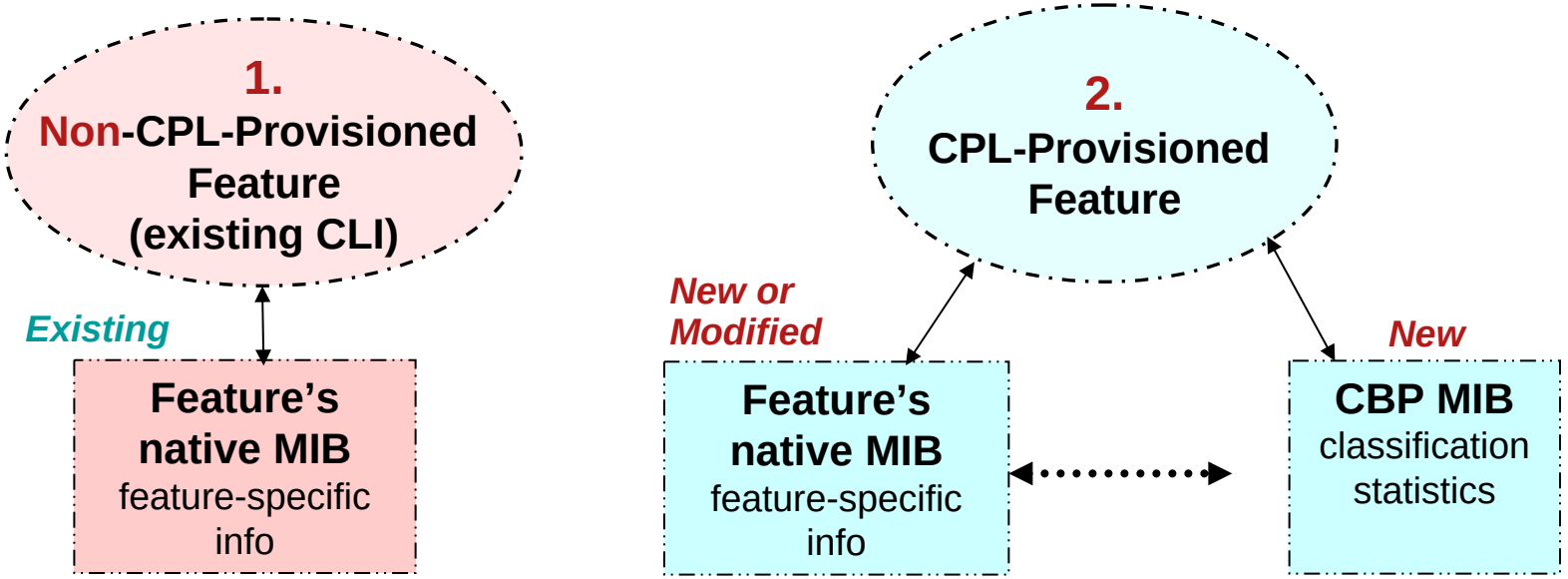
# Feature MIBs and CBP MIB

- CPL-provisioned features will link to two MIBs
  - Their own feature-specific MIB
  - The CBP MIB



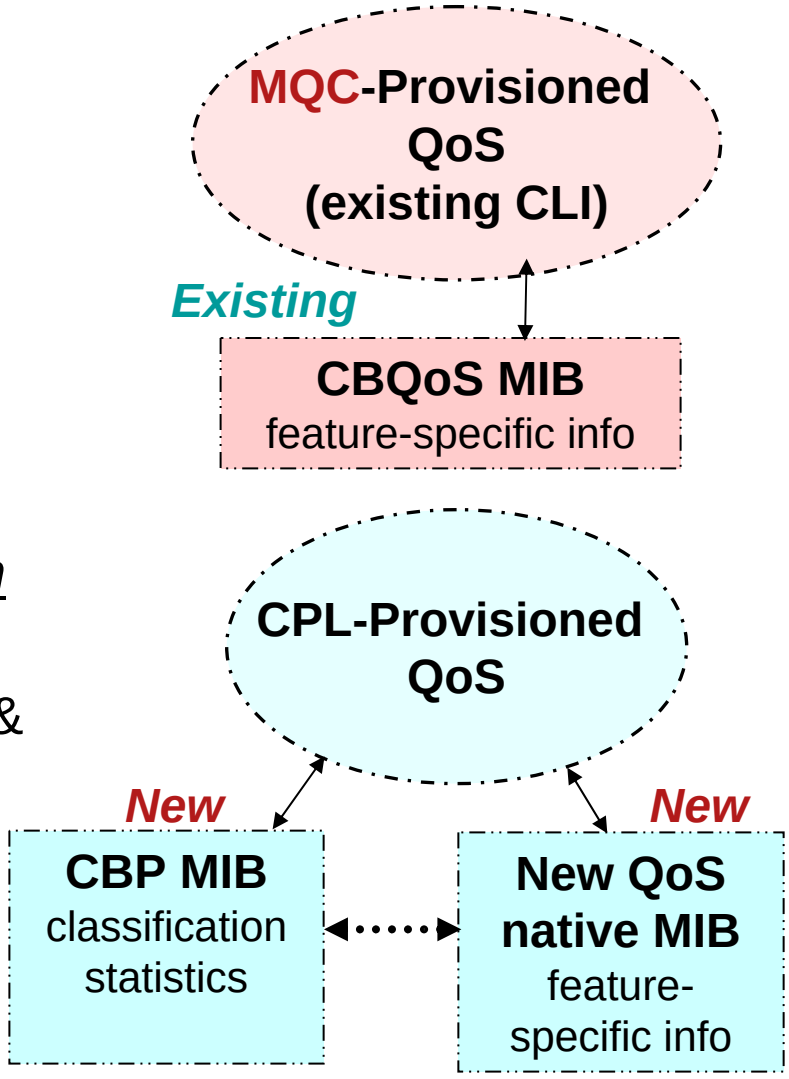
# Mirrors Feature Adoption of CPL

- A MIB often reflects a feature's provisioning syntax
  - The CBP MIB reflects the class-based provisioning model of CPL
  - Information that is common to multiple features must be accessible through a common MIB
- Provisioning method and feature MIBs
  1. Non-CPL Provisioned: Use existing feature MIB
  2. CPL-Provisioned: Use CBP MIB with feature-specific MIB



# Quality of Service: Evolution to CPL from MQC

- MQC is a proper subset of CPL
- Existing MQC configurations are forward-compatible to CPL
- *Router does not distinguish between CPL and MQC*
  - Common statistics are counted in *both* CBQoS MIB and new CBP MIB
  - Use one (CBQoS) *or* the other (CBP & new QoS)
  - CBQoS MIB will be supported indefinitely



# ROADMAP CLASS-BASED POLICY PROVISIONING

# Class-Based Policy Provisioning Roadmap – IOS Release 12.4T

## Features Using CPL

| Release 12.4T | | Description |
|---|---|---|
| -- | **Pre-12.4T** | ▪ Quality of Service (QoS)<br>▪ Control Plane Protection (CPPr) |
| 2nd | 12.4(4)T<br>**11/14/2005** | ▪ Flexible Packet Matching (FPM) |
| 3rd | **02/2006** | ▪ IOS Firewall |
| 4th | **05/2006** | ▪ FPM with CPL-XML<br>▪ IP SLAs |
| 6th | **2H 2006** | ▪ NetFlow<br>▪ Intrusion Prevention System (IPS) |

## Platform Support

| | |
|---|---|
| Routers | 800 Series, 1700 Series, 1800 Series, 2691, 2600XM Series, 2800 Series, 3700 Series, 3800 Series, 7200 Series, 7301 |

# QUESTIONS AND ANSWERS

# Q&A – 1

- Will existing MQC configurations still work?

  Yes. MQC syntax is a proper subset of CPL. Existing MQC configurations are forward-compatible with CPL.

- Will there be a “type” keyword for QoS? Will existing MQC configurations convert to it?

  Not at this time. Future evolution will determine the need. For the foreseeable future, QoS (MQC) configurations remain untyped.

- Will there be a tool to convert old style CLI to CPL syntax?

  There are no plans for a syntax conversion tool. Future evolution will determine the need. It is possible that some features may have a higher need than others and feature-specific converters may emerge.

# Q&A – 2

- What is the order of operations when multiple CPL-configured polices are on an interface?

  CPL provisioning does not affect order of operations in the feature path.