



有線ネットワークでの **Application Visibility and Control** の設定

- [有線ネットワークでの Application Visibility and Control について \(1 ページ\)](#)
- [サポートされる AVC クラス マップおよびポリシー マップのフォーマット \(2 ページ\)](#)
- [有線 Application Visibility and Control の制限 \(3 ページ\)](#)
- [Application Visibility and Control の設定方法 \(5 ページ\)](#)
- [Application Visibility and Control のモニタリング \(33 ページ\)](#)
- [例：Application Visibility and Control の設定 \(33 ページ\)](#)
- [基本的なトラブルシューティング：質問と回答 \(45 ページ\)](#)
- [Application Visibility and Control に関する追加情報 \(47 ページ\)](#)
- [有線ネットワークでの Application Visibility and Control の機能履歴 \(47 ページ\)](#)

有線ネットワークでの **Application Visibility and Control** について

Application Visibility and Control (AVC) は、アプリケーションへの適応力やアプリケーションへのインテリジェンス性に基づいて、厳密なパケットおよび接続からブランチおよびキャンパスソリューションを発展させるためのシスコの取り組みの重要な部分です。Application Visibility and Control (AVC) は、ネットワークベースのアプリケーション認識 (NBAR2) エンジンによるディープパケットインスペクション技術を使用してアプリケーションを分類します。AVC は、スタンドアロンスイッチおよびスイッチスタックの有線アクセスポート上に設定できます。NBAR2 は、プロトコル検出を有効にすることによって明示的に、または **match protocol** 分類子を含む QoS ポリシーを接続することによって暗黙的に、インターフェイス上でアクティブにできます。有線 AVC Flexible Netflow (FNF) をインターフェイス上に設定し、インターフェイスごとのクライアント、サーバ、アプリケーションの統計情報を提供できます。このレコードは、Easy Performance Monitor (Easy perf-mon または ezPM) の **application-statistics** および **application-performance** プロファイルで利用できる **application-client-server-stats** トラフィック監視と同様です。

サポートされる AVC クラス マップおよびポリシー マップのフォーマット

ここでは、サポートされている AVC クラスマップとポリシーマップ形式について説明します。

サポートされる AVC クラス マップのフォーマット

クラスマップのフォーマット	クラスマップの例	方向
<code>match protocol protocol name</code>	<code>class-map match-any NBAR-VOICE match protocol ms-lync-audio</code>	入力と出力の両方
組み合わせフィルタ	<code>class-map match-any NBAR-VOICE match protocol ms-lync-audio match dscp ef</code>	入力と出力の両方

サポートされる AVC ポリシーのフォーマット

ポリシーのフォーマット	QoS 処理
<code>match protocol</code> フィルタに基づく出力ポリシー	マークおよびポリシー
<code>match protocol</code> フィルタに基づく入力ポリシー	マークおよびポリシー

次の表で、AVC ポリシーの詳細なフォーマット、および例について説明します。

AVC ポリシーのフォーマット	AVC ポリシーの例	方向
ベーシック セット	<code>policy-map MARKING-IN class NBAR-MM_CONFERENCEING set dscp af41</code>	入力および出力
ベーシック ポリシー	<code>policy-map POLICING-IN class NBAR-MM_CONFERENCEING police cir 600000 set dscp af41</code>	入力および出力
ベーシック セットおよびポリシー	<code>policy-map webex-policy class webex-class set dscp ef police 5000000</code>	入力および出力

AVC ポリシーのフォーマット	AVC ポリシーの例	方向
デフォルトを含む複数のセットおよびポリシー	<pre> policy-map webex-policy class webex-class set dscp af31 police 4000000 class class-webex-category set dscp ef police 6000000 class class-default set dscp <> </pre>	入力および出力
階層型ポリシー	<pre> policy-map webex-policy class webex-class police 5000000 service-policy client-in-police-only policy-map client-in-police-only class webex-class police 100000 class class-webex-category set dscp ef police 200000 </pre>	入力および出力
階層型セットおよびポリシー	<pre> policy-map webex-policy class class-default police 1500000 service policy client-up-child policy-map client-up-child class webex-class police 100000 set dscp ef class class-webex-category police 200000 set dscp af31 </pre>	

有線 Application Visibility and Control の制限

- NBAR 対応 QoS ポリシー設定は有線物理ポートでのみ許可されます。ポリシー設定は、VLANおよびその他の論理インターフェイスなどの仮想インターフェイスではサポートされていません。
- NBAR ベースの QoS ポリシー設定は、ポートチャネルメンバーポートおよび SVI やサブインターフェイスなどの仮想インターフェイスではサポートされません。
- NBAR ベースの QoS ポリシー設定は、レイヤ 2 アクセスポートとトランクポート、およびレイヤ 3 ルーテッドポートでサポートされます。
- NBAR と送信 (Tx) スイッチドポートアナライザ (SPAN) は、同じインターフェイスではサポートされません。

- プロトコルベースまたは属性ベースのいずれかのポートに同時に接続できるのは、NBAR ベースの QoS メカニズムの 1 つだけです。次の 2 つの属性のみがサポートされます。
 - traffic-class
 - business-relevance
- 従来の WDAVC QoS の制限事項は引き続き適用されます。
 - マーキングとポリシングのみがサポートされます。
 - 物理インターフェイスだけがサポートされます。
 - アプリケーション分類がオフラインで行われるため、QoS 分類には遅延があります (ただし、フローの最初のパケットは、正確な QoS 分類の前に転送されます)。
- NBAR2 ベースの一致基準 **match protocol** は、マーキングアクションおよびポリシングアクションでのみ許可されます。NBAR2 一致基準は、キューイング機能が設定されているポリシーでは許可されません。
- 「一致プロトコル」：すべてのポリシーで最大 255 の同時に異なるプロトコル (8 ビットの HW 制限)。
- AVC は管理ポート (Gig 0/0) ではサポートされていません。
- IPv6 パケットの分類はサポートされていません。
- IPv4 ユニキャスト (TCP/UDP) のみがサポートされます。
- Web UI : Web UI からアプリケーションの可視性を設定し、アプリケーションのモニタリングを実行できます。アプリケーション制御は、CLI を使用してのみ実行できます。Web UI ではサポートされていません。

Web UI 上で有線 AVC のトラフィックを管理、またはチェックするには、最初に CLI を使用して **ip http authentication local** と **ip nbar http-service** コマンドを設定する必要があります。
- NBAR および ACL のロギングは、同一スイッチ上で一緒に設定することはできません。
- プロトコル検出、アプリケーションベースの QoS、および有線 AVC FNF は、非アプリケーションベース FNF がある同一インターフェイス上で同時に設定することはできません。ただし、これらの有線 AVC 機能は、相互に設定できます。たとえば、プロトコル検出、アプリケーションベースの QoS、および有線 AVC FNF は、同一インターフェイス上で同時に設定できます。
- それぞれ異なる定義済みレコードを持つ 2 つの有線 AVC モニタのみをインターフェイスに同時に接続できます。
- 2 つの方向性フローレコード (入力と出力) と 2 つの従来のフローレコードがサポートされます。

- 接続は、物理 Layer2（アクセス/トランク）および Layer3 ポートでのみ行う必要があります。アップリンクは、単一のアップリンクであり、ポートチャネルの一部でなければ接続できません。
- パフォーマンス：各スイッチ メンバーは、50% 未満の CPU 使用率で、1 秒あたり 500 の接続（CPS）を処理できます。
- 拡張性：24 個のアクセスポートと 48 個のアクセスポートごとに最大 5000 の双方向フローを処理できます。
- 有線 AVC では、この章の手順にリストされている固定のフィールドセットのみを使用できます。その他の組み合わせは使用できません。通常の FNF フローモニタでは、他の組み合わせも使用できます（サポートされている FNF フィールドのリストについては、『*Network Management Configuration Guide*』の「Configuring Flexible NetFlow」の章を参照してください）。

Application Visibility and Control の設定方法

有線ネットワークでの Application Visibility and Control の設定

有線ポートで Application Visibility and Control を設定するには、次の手順を実行します。

可視性の設定

- インターフェイス コンフィギュレーション モードで **ip nbar protocol-discovery** コマンドを使用してインターフェイス上でプロトコル検出を有効にすることで、NBAR2 エンジン をアクティブ化します。[インターフェイスでのアプリケーション認識の有効化（6 ページ）](#) を参照してください。

制御設定：次の手順に従って、アプリケーションに基づいて QoS ポリシーを設定します。

1. AVC QoS ポリシーの作成。[AVC QoS ポリシーの作成（6 ページ）](#) を参照してください。
2. インターフェイスへの AVC QoS ポリシーの適用。[スイッチポートへの QoS ポリシーの適用（9 ページ）](#) を参照してください。

アプリケーション ベースの Flexible Netflow の設定：

- フローにキー フィールドおよび非キー フィールドを指定して、フロー レコードを作成します。[フロー レコードの作成（10 ページ）](#) を参照してください。
- フローエクスポートを作成してフローレコードをエクスポートします。[フローエクスポートの作成（24 ページ）](#) を参照してください。
- フロー レコードおよびフロー エクスポートに基づいて、フロー モニタを作成します。[フロー モニタの作成（25 ページ）](#) を参照してください。

- インターフェイスにフロー モニタを接続します。 [インターフェイスへのフロー モニタの関連付け \(27 ページ\)](#) を参照してください。

プロトコル検出、アプリケーションベースの QoS およびアプリケーションベースの FNF は、すべて独立した機能です。単独で設定することも、または同じインターフェイスで同時に設定することもできます。

インターフェイスでのアプリケーション認識の有効化

インターフェイス上でアプリケーション認識をイネーブルにするには、次の手順を実行します。

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	interface interface-id 例： Device(config)# interface gigabitethernet 1/0/1	プロトコル検出をイネーブルにするインターフェイスを指定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 3	ip nbar protocol-discovery 例： Device(config-if)# ip nbar protocol-discovery	NBAR2 エンジンを実アクティブ化することで、インターフェイスでアプリケーション認識を有効にします。
ステップ 4	end 例： Device(config-if)# end	特権 EXEC モードに戻ります。

AVC QoS ポリシーの作成

AVC QoS ポリシーを作成するには、次の一般的な手順を実行します。

1. match protocol フィルタでクラス マップを作成します。
2. ポリシー マップを作成します。

3. インターフェイスにポリシー マップを適用します。

クラス マップの作成

match protocol フィルタを設定する前に、クラス マップを作成する必要があります。マーキングやポリシングなどの QoS アクションをトラフィックに適用できます。AVC の match protocol フィルタは、有線アクセスポートに適用されます。サポートされているプロトコルの詳細については、http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/prot_lib/config_library/nbar-prot-pack-library.html を参照してください。

手順

	コマンドまたはアクション	目的
ステップ 1	terminal 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	class-map class-map-name 例： Device(config)# class-map webex-class	クラス マップを作成します。
ステップ 3	match protocol application-name 例： Device(config)# class-map webex-class Device(config-cmap)# match protocol webex-media	アプリケーション名との一致を指定します。
ステップ 4	end 例： Device(config)# end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバル コンフィギュレーション モードを終了できます。

ポリシー マップの作成

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	policy-map policy-map-name 例：	ポリシーマップ名を入力することによってポリシー マップを作成し、ポリシー

	コマンドまたはアクション	目的
	Device(config)# policy-map webex-policy	<p>マップ コンフィギュレーション モードを開始します。</p> <p>デフォルトでは、ポリシー マップは定義されていません。</p> <p>ポリシー マップのデフォルトの動作では、パケットが IP パケットの場合は DSCP が 0 に、パケットがタグ付きの場合は CoS が 0 に設定されます。ポリシー マップは実行されません。</p> <p>(注) 既存のポリシーマップを削除するには、no policy-map policy-map-name グローバル コンフィギュレーション コマンドを使用します。</p>
ステップ 3	<p>class [<i>class-map-name</i> class-default]</p> <p>例 :</p> <p>Device(config-pmap)# class webex-class</p>	<p>トラフィックの分類を定義し、ポリシー マップ クラス コンフィギュレーション モードを開始します。</p> <p>デフォルトでは、ポリシー マップおよびクラスマップは定義されていません。</p> <p>すでに class-map グローバル コンフィギュレーション コマンドを使用してトラフィッククラスが定義されている場合は、このコマンドで <i>class-map-name</i> にその名前を指定します。</p> <p>class-default トラフィッククラスは定義済みで、どのポリシーにも追加できます。このトラフィック クラスは、常にポリシーマップの最後に配置されます。暗黙の match any が class-default クラスに含まれている場合、他のトラフィック クラスと一致しないパケットはすべて class-default と一致します。</p> <p>(注) 既存のクラスマップを削除するには、no class class-map-name ポリシーマップ コンフィギュレーション コマンドを使用します。</p>

	コマンドまたはアクション	目的
ステップ 4	police rate-bps burst-byte 例 : Device(config-pmap-c) # police 100000 80000	分類したトラフィックにポリサーを定義します。 デフォルトでは、ポリサーは定義されていません。 <ul style="list-style-type: none"> • <i>rate-bps</i> には、平均トラフィックレートをビット/秒 (bps) で指定します。指定できる範囲は 8000 ~ 10000000000 です • <i>burst-byte</i> には、標準バーストサイズをバイト数で指定します。有効範囲は、1000 ~ 512000000 です。
ステップ 5	set {dscp new-dscp cos cos-value} 例 : Device(config-pmap-c) # set dscp 45	パケットに新しい値を設定することによって、IP トラフィックを分類します。 <ul style="list-style-type: none"> • dscp new-dscp には、分類されたトラフィックに割り当てる新しい DSCP 値を入力します。指定できる範囲は 0 ~ 63 です。
ステップ 6	end 例 : Device(config) # end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコンフィギュレーションモードを終了できます。

スイッチポートへの QoS ポリシーの適用

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 : Device# configure terminal	グローバル コンフィギュレーションモードを開始します。
ステップ 2	interface interface-id 例 : Device(config) # interface Gigabitethernet 1/0/1	インターフェイスコンフィギュレーションモードを開始します。

	コマンドまたはアクション	目的
ステップ 3	service-policy input <i>polycymapname</i> 例： Device(config-if)# service-policy input MARKING_IN	インターフェイスにローカル ポリシーを適用します。
ステップ 4	end 例： Device(config)# end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバル コンフィギュレーション モードを終了できます。

有線 AVC Flexible Netflow の設定

フロー レコードの作成

有線 AVC FNF は、従来の双方向フローレコードと方向性フローレコード（入力と出力）の 2 種類の定義済みフローレコードをサポートします。合計 4 つの異なる定義済みフローレコード（2 つの双方向フローレコードと 2 つの方向性フローレコード）を設定し、フローモニタに関連付けることができます。従来の双方向レコードはクライアント/サーバアプリケーション統計情報レコードであり、新しい方向性レコードは入出力のアプリケーション統計情報です。

双方向フローレコード

フローレコード 1：双方向フローレコード

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	flow record <i>flow_record_name</i> 例： Device(config)# flow record fr-wdavic-1	フローレコードコンフィギュレーション モードを開始します。
ステップ 3	description <i>description</i> 例： Device(config-flow-record)# description fr-wdavic-1	(任意) フローレコードの説明を作成します。
ステップ 4	match ipv4 version 例： Device(config-flow-record)# match ipv4 version	IPv4 ヘッダーからの IP バージョンとの一致を指定します。

	コマンドまたはアクション	目的
ステップ 5	match ipv4 protocol 例 : Device (config-flow-record) # match ipv4 protocol	IPv4 プロトコルとの一致を指定します。
ステップ 6	match application name 例 : Device (config-flow-record) # match application name	アプリケーション名との一致を指定します。 (注) この操作は、AVC サポートでは必須です。フローがアプリケーションと一致することが可能になるためです。
ステップ 7	match connection client ipv4 address 例 : Device (config-flow-record) # match connection client ipv4 address	クライアント (フローイニシエータ) の IPv4 アドレスとの一致を指定します。
ステップ 8	match connection server ipv4 address 例 : Device (config-flow-record) # match connection server ipv4 address	サーバ (フローレスポンド) の IPv4 アドレスとの一致を指定します。
ステップ 9	match connection server transport port 例 : Device (config-flow-record) # match connection server transport port	サーバのポート番号との一致を指定します。
ステップ 10	match flow observation point 例 : Device (config-flow-record) # match flow observation point	フロー観測メトリックの観測ポイント ID との一致を指定します。
ステップ 11	collect flow direction 例 : Device (config-flow-record) # collect flow direction	次の手順で collect connection initiator コマンドの initiator キーワードで指定される双方向フローの関連する側 (イニシエータまたはレスポンド) の方向 (入力または出力) を収集するように指定します。 initiator キーワードで指定される値に応じて、 flow direction キーワードは次の値をとります。 <ul style="list-style-type: none"> • 0x01 = 入力フロー • 0x02 = 出力フロー

	コマンドまたはアクション	目的
		initiator キーワードがイニシエータに設定されている場合、フローの方向はフローのイニシエータ側から指定されます。 initiator キーワードがレスポンドに設定されている場合、フローの方向はフローのレスポンド側から指定されます。有線 AVC では、 initiator キーワードは常にイニシエータに設定されています。
ステップ 12	collect connection initiator 例 : Device(config-flow-record) # collect connection initiator	collect flow direction コマンドで指定されたフローの方向に関連するフローの側（イニシエータまたはレスポンド）を収集するように指定します。 initiator キーワードは、フローの方向に関する次の情報を提供します。 <ul style="list-style-type: none"> • 0x01 = イニシエータ : フローの送信元は接続のイニシエータです 有線 AVC では、 initiator キーワードは常にイニシエータに設定されています。
ステップ 13	collect connection new-connections 例 : Device(config-flow-record) # collect connection new-connections	観測された接続開始の数を収集するように指定します。
ステップ 14	collect connection client counter packets long 例 : Device(config-flow-record) # collect connection client counter packets long	クライアントが送信したパケット数を収集するように指定します。
ステップ 15	collect connection client counter bytes network long 例 : Device(config-flow-record) # collect connection client counter bytes network long	クライアントが送信したバイト数の合計を収集するように指定します。
ステップ 16	collect connection server counter packets long 例 :	サーバが送信したパケット数を収集するように指定します。

	コマンドまたはアクション	目的
	<code>Device (config-flow-record) # collect connection server counter packets long</code>	
ステップ 17	collect connection server counter bytes network long 例： <code>Device (config-flow-record) # collect connection server counter bytes network long</code>	サーバが送信したバイト数の合計を収集するように指定します。
ステップ 18	collect timestamp absolute first 例： <code>Device (config-flow-record) # collect timestamp absolute first</code>	最初のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 19	collect timestamp absolute last 例： <code>Device (config-flow-record) # collect timestamp absolute last</code>	最新のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 20	end 例： <code>Device (config) # end</code>	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコンフィギュレーションモードを終了できます。
ステップ 21	show flow record 例： <code>Device # show flow record</code>	すべてのフローレコードに関する情報を表示します。

フローレコード 2: 双方向フローレコード

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： <code>Device # configure terminal</code>	グローバル コンフィギュレーションモードを開始します。
ステップ 2	flow record <i>flow_record_name</i> 例： <code>Device (config) # flow record fr-wdavic-1</code>	フローレコードコンフィギュレーションモードを開始します。
ステップ 3	description <i>description</i> 例：	(任意) フローレコードの説明を作成します。

	コマンドまたはアクション	目的
	Device(config-flow-record) # description fr-wdavic-1	
ステップ 4	match ipv4 version 例 : Device(config-flow-record) # match ipv4 version	IPv4 ヘッダーからの IP バージョンとの一致を指定します。
ステップ 5	match ipv4 protocol 例 : Device(config-flow-record) # match ipv4 protocol	IPv4 プロトコルとの一致を指定します。
ステップ 6	match application name 例 : Device(config-flow-record) # match application name	アプリケーション名との一致を指定します。 (注) この操作は、AVC サポートでは必須です。フローがアプリケーションと一致することが可能になるためです。
ステップ 7	match connection client ipv4 address 例 : Device(config-flow-record) # match connection client ipv4 address	クライアント (フローイニシエータ) の IPv4 アドレスとの一致を指定します。
ステップ 8	match connection client transport port 例 : Device(config-flow-record) # match connection client transport port	(任意) フローレコードのキーフィールドとして、クライアントの接続ポートとの一致を指定します。
ステップ 9	match connection server ipv4 address 例 : Device(config-flow-record) # match connection server ipv4 address	サーバ (フローレスポンド) の IPv4 アドレスとの一致を指定します。
ステップ 10	match connection server transport port 例 : Device(config-flow-record) # match connection server transport port	サーバのトランスポートポートとの一致を指定します。
ステップ 11	match flow observation point 例 : Device(config-flow-record) # match flow observation point	フロー観測メトリックの観測ポイント ID との一致を指定します。

	コマンドまたはアクション	目的
ステップ 12	collect flow direction 例 : Device(config-flow-record)# collect flow direction	次の手順で collect connection initiator コマンドの initiator キーワードで指定される双方向フローの関連する側（イニシエータまたはレスポンド）の方向（入力または出力）を収集するように指定します。 initiator キーワードで指定される値に応じて、 flow direction キーワードは次の値をとります。 <ul style="list-style-type: none"> • 0x01 = 入力フロー • 0x02 = 出力フロー initiator キーワードがイニシエータに設定されている場合、フローの方向はフローのイニシエータ側から指定されます。 initiator キーワードがレスポンドに設定されている場合、フローの方向はフローのレスポンド側から指定されます。有線 AVC では、 initiator キーワードは常にイニシエータに設定されています。
ステップ 13	collect connection initiator 例 : Device(config-flow-record)# collect connection initiator	collect flow direction コマンドで指定されたフローの方向に関連するフローの側（イニシエータまたはレスポンド）を収集するように指定します。 initiator キーワードは、フローの方向に関する次の情報を提供します。 <ul style="list-style-type: none"> • 0x01 = イニシエータ：フローの送信元は接続のイニシエータです 有線 AVC では、 initiator キーワードは常にイニシエータに設定されています。
ステップ 14	collect connection new-connections 例 : Device(config-flow-record)# collect connection new-connections	観測された接続開始の数を収集するように指定します。
ステップ 15	collect connection client counter packets long 例 :	クライアントが送信したパケット数を収集するように指定します。

	コマンドまたはアクション	目的
	<code>Device(config-flow-record)# collect connection client counter packets long</code>	
ステップ 16	collect connection client counter bytes network long 例： <code>Device(config-flow-record)# collect connection client counter bytes network long</code>	クライアントが送信したバイト数の合計を収集するように指定します。
ステップ 17	collect connection server counter packets long 例： <code>Device(config-flow-record)# collect connection server counter packets long</code>	サーバが送信したパケット数を収集するように指定します。
ステップ 18	collect connection server counter bytes network long 例： <code>Device(config-flow-record)# collect connection server counter bytes network long</code>	サーバが送信したバイト数の合計を収集するように指定します。
ステップ 19	collect timestamp absolute first 例： <code>Device(config-flow-record)# collect timestamp absolute first</code>	最初のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 20	collect timestamp absolute last 例： <code>Device(config-flow-record)# collect timestamp absolute last</code>	最新のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 21	end 例： <code>Device(config)# end</code>	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコンフィギュレーションモードを終了できます。
ステップ 22	show flow record 例： <code>Device# show flow record</code>	すべてのフローレコードに関する情報を表示します。

方向性フローレコード

フローレコード 3 : 方向性フローレコード : 入力

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 : Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	flow record flow_record_name 例 : Device(config)# flow record fr-wdavic-3	フローレコードコンフィギュレーション モードを開始します。
ステップ 3	description description 例 : Device(config-flow-record)# description flow-record-1	(任意) フローレコードの説明を作成します。
ステップ 4	match ipv4 version 例 : Device(config-flow-record)# match ipv4 version	IPv4 ヘッダーからの IP バージョンとの一致を指定します。
ステップ 5	match ipv4 protocol 例 : Device(config-flow-record)# match ipv4 protocol	IPv4 プロトコルとの一致を指定します。
ステップ 6	match ipv4 source address 例 : Device(config-flow-record)# match ipv4 source address	IPv4 送信元アドレスとの一致をキーフィールドとして指定します。
ステップ 7	match ipv4 destination address 例 : Device(config-flow-record)# match ipv4 destination address	IPv4 宛先アドレスとの一致をキーフィールドとして指定します。
ステップ 8	match transport source-port 例 : Device(config-flow-record)# match transport source-port	トランスポート発信元ポートとの一致をキーフィールドとして指定します。
ステップ 9	match transport destination-port 例 : Device(config-flow-record)# match transport destination-port	トランスポート宛先ポートとの一致をキーフィールドとして指定します。

	コマンドまたはアクション	目的
ステップ 10	match interface input 例 : Device(config-flow-record) # match interface input	入力インターフェイスとの一致をキーフィールドとして指定します。
ステップ 11	match application name 例 : Device(config-flow-record) # match application name	アプリケーション名との一致を指定します。 (注) この操作は、AVC サポートでは必須です。フローがアプリケーションと一致することが可能になるためです。
ステップ 12	collect interface output 例 : Device(config-flow-record) # collect interface output	フローから出力インターフェイスを収集するように指定します。
ステップ 13	collect counter bytes long 例 : Device(config-flow-record) # collect counter bytes long	フローのバイト数を収集するように指定します。
ステップ 14	collect counter packets long 例 : Device(config-flow-record) # collect counter packets long	フローのパケット数を収集するように指定します。
ステップ 15	collect timestamp absolute first 例 : Device(config-flow-record) # collect timestamp absolute first	最初のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 16	collect timestamp absolute last 例 : Device(config-flow-record) # collect timestamp absolute last	最新のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 17	end 例 : Device(config) # end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコンフィギュレーションモードを終了できます。
ステップ 18	show flow record 例 :	すべてのフローレコードに関する情報を表示します。

	コマンドまたはアクション	目的
	Device# show flow record	

フローレコード 4 : 方向性フローレコード : 出力

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 : Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	flow record <i>flow_record_name</i> 例 : Device (config)# flow record fr-wdavic-4	フローレコードコンフィギュレーション モードを開始します。
ステップ 3	description <i>description</i> 例 : Device (config-flow-record)# description flow-record-1	(任意) フローレコードの説明を作成します。
ステップ 4	match ipv4 version 例 : Device (config-flow-record)# match ipv4 version	IPv4 ヘッダーからの IP バージョンとの一致を指定します。
ステップ 5	match ipv4 protocol 例 : Device (config-flow-record)# match ipv4 protocol	IPv4 プロトコルとの一致を指定します。
ステップ 6	match ipv4 source address 例 : Device (config-flow-record)# match ipv4 source address	IPv4 送信元アドレスとの一致をキーフィールドとして指定します。
ステップ 7	match ipv4 destination address 例 : Device (config-flow-record)# match ipv4 destination address	IPv4 宛先アドレスとの一致をキーフィールドとして指定します。
ステップ 8	match transport source-port 例 : Device (config-flow-record)# match transport source-port	トランスポート発信元ポートとの一致をキーフィールドとして指定します。

	コマンドまたはアクション	目的
ステップ 9	match transport destination-port 例 : Device(config-flow-record) # match transport destination-port	トランスポート宛先ポートとの一致をキーフィールドとして指定します。
ステップ 10	match interface output 例 : Device(config-flow-record) # match interface output	出力インターフェイスとの一致をキーフィールドとして指定します。
ステップ 11	match application name 例 : Device(config-flow-record) # match application name	アプリケーション名との一致を指定します。 (注) この操作は、AVC サポートでは必須です。フローがアプリケーションと一致することが可能になるためです。
ステップ 12	collect interface input 例 : Device(config-flow-record) # collect interface input	フローから入力インターフェイスを収集するように指定します。
ステップ 13	collect counter bytes long 例 : Device(config-flow-record) # collect counter bytes long	フローのバイト数を収集するように指定します。
ステップ 14	collect counter packets long 例 : Device(config-flow-record) # collect counter packets long	フローのパケット数を収集するように指定します。
ステップ 15	collect timestamp absolute first 例 : Device(config-flow-record) # collect timestamp absolute first	最初のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 16	collect timestamp absolute last 例 : Device(config-flow-record) # collect timestamp absolute last	最新のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 17	end 例 :	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコ

	コマンドまたはアクション	目的
	Device (config) # end	ンフィギュレーションモードを終了できます。
ステップ 18	show flow record 例 : Device# show flow record	すべてのフローレコードに関する情報を表示します。

DNS フローレコード

フローレコード 5 : DNS フローレコード

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 : Device# configure terminal	グローバル コンフィギュレーションモードを開始します。
ステップ 2	flow record <i>flow_record_name</i> 例 : Device (config) # flow record fr-wdavic-5	フローレコードコンフィギュレーションモードを開始します。
ステップ 3	description <i>description</i> 例 : Device (config-flow-record) # description flow-record-5	(任意) フローレコードの説明を作成します。
ステップ 4	match ipv4 version 例 : Device (config-flow-record) # match ipv4 version	IPv4 ヘッダーからの IP バージョンとの一致を指定します。
ステップ 5	match ipv4 protocol 例 : Device (config-flow-record) # match ipv4 protocol	IPv4 プロトコルとの一致を指定します。
ステップ 6	match application name 例 : Device (config-flow-record) # match application name	アプリケーション名との一致を指定します。 (注) この操作は、AVC サポートでは必須です。フローがアプリケーションと一致することが可能になるためです。

	コマンドまたはアクション	目的
ステップ 7	match connection client ipv4 address 例： Device(config-flow-record) # match connection client ipv4 address	クライアント（フローイニシエータ）の IPv4 アドレスとの一致を指定します。
ステップ 8	match connection client transport port 例： Device(config-flow-record) # match connection client transport port	フローレコードのキーフィールドとして、クライアントの接続ポートとの一致を指定します。
ステップ 9	match connection server ipv4 address 例： Device(config-flow-record) # match connection server ipv4 address	サーバ（フローレスポンド）の IPv4 アドレスとの一致を指定します。
ステップ 10	match connection server transport port 例： Device(config-flow-record) # match connection server transport port	サーバのポート番号との一致を指定します。
ステップ 11	collect flow direction 例： Device(config-flow-record) # collect flow direction	<p>次の手順で collect connection initiator コマンドの initiator キーワードで指定される双方向フローの関連する側（イニシエータまたはレスポンド）の方向（入力または出力）を収集するように指定します。initiator キーワードで指定される値に応じて、flow direction キーワードは次の値をとります。</p> <ul style="list-style-type: none"> • 0x01 = 入力フロー • 0x02 = 出力フロー <p>initiator キーワードがイニシエータに設定されている場合、フローの方向はフローのイニシエータ側から指定されます。initiator キーワードがレスポンドに設定されている場合、フローの方向はフローのレスポンド側から指定されます。有線 AVC では、initiator キーワードは常にイニシエータに設定されています。</p>

	コマンドまたはアクション	目的
ステップ 12	collect timestamp absolute first 例 : Device(config-flow-record)# collect timestamp absolute first	最初のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 13	collect timestamp absolute last 例 : Device(config-flow-record)# collect timestamp absolute last	最新のパケットがフローで確認されたときの時間をミリ秒単位で収集するように指定します。
ステップ 14	collect connection initiator 例 : Device(config-flow-record)# collect connection initiator	collect flow direction コマンドで指定されたフローの方向に関連するフローの側（イニシエータまたはレスポнда）を収集するように指定します。 initiator キーワードは、フローの方向に関する次の情報を提供します。 <ul style="list-style-type: none"> • 0x01 = イニシエータ : フローの送信元は接続のイニシエータです 有線 AVC では、 initiator キーワードは常にイニシエータに設定されています。
ステップ 15	collect connection new-connections 例 : Device(config-flow-record)# collect connection new-connections	観測された接続開始の数を収集するように指定します。
ステップ 16	collect connection server counter packets long 例 : Device(config-flow-record)# collect connection server counter packets long	サーバが送信したパケット数を収集するように指定します。
ステップ 17	collect connection client counter packets long 例 : Device(config-flow-record)# collect connection client counter packets long	クライアントが送信したパケット数を収集するように指定します。
ステップ 18	collect connection server counter bytes network long 例 :	サーバが送信したバイト数の合計を収集するように指定します。

	コマンドまたはアクション	目的
	<code>Device(config-flow-record)# collect connection server counter bytes network long</code>	
ステップ 19	collect connection client counter bytes network long 例： <code>Device(config-flow-record)# collect connection client counter bytes network long</code>	クライアントが送信したバイト数の合計を収集するように指定します。
ステップ 20	collect application dns domain-name 例： <code>Device(config-flow-record)# collect application dns domain-name</code>	DNS ドメイン名を DNS フローレコードの収集フィールドとして使用するよう設定します。
ステップ 21	end 例： <code>Device(config)# end</code>	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバル コンフィギュレーションモードを終了できます。

フロー エクスポートの作成

フロー エクスポートを作成すると、フローのエクスポート パラメータを定義できます。

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： <code>Device# configure terminal</code>	グローバル コンフィギュレーション モードを開始します。
ステップ 2	flow exporter flow_exporter_name 例： <code>Device(config)# flow exporter flow-exporter-1</code>	フロー エクスポート コンフィギュレーション モードを開始します。
ステップ 3	description description 例： <code>Device(config-flow-exporter)# description flow-exporter-1</code>	(任意) フロー エクスポートの説明を作成します。
ステップ 4	destination { hostname ipv4-address ipv6-address } 例：	エクスポートでデータを送信する宛先システムのホスト名、IPv4 または IPv6 アドレスを指定します。

	コマンドまたはアクション	目的
	Device(config-flow-exporter)# destination 10.10.1.1	
ステップ 5	option application-table [timeout seconds] 例： Device(config-flow-exporter)# option application-table timeout 500	(任意) フロー エクスポートのアプリケーション テーブルのオプションを設定します。 timeout オプションを使用すると、フローエクスポートの再送信時間を秒単位で設定できます。有効な範囲は 1 ~ 86400 秒です。
ステップ 6	end 例： Device(config)# end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバル コンフィギュレーション モードを終了できます。
ステップ 7	show flow exporter 例： Device# show flow exporter	すべてのフロー エクスポートに関する情報を表示します。
ステップ 8	show flow exporter statistics 例： Device# show flow exporter statistics	フロー エクスポートの統計情報を表示します。

フロー モニタの作成

フロー モニタを作成して、フロー レコードに関連付けることができます。

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 2	flow monitor monitor-name 例： Device(config)# flow monitor flow-monitor-1	フロー モニタを作成し、フロー モニタ コンフィギュレーション モードを開始します。
ステップ 3	description description 例： Device(config-flow-monitor)# description flow-monitor-1	(任意) フロー モニタの説明を作成します。

	コマンドまたはアクション	目的
ステップ 4	record <i>record-name</i> 例： Device(config-flow-monitor)# record flow-record-1	事前に作成されたレコードの名前を指定します。
ステップ 5	exporter <i>exporter-name</i> 例： Device(config-flow-monitor)# exporter flow-exporter-1	事前に作成されたエクスポートの名前を指定します。
ステップ 6	cache { entries <i>number-of-entries</i> timeout { active inactive } type normal } 例： Device(config-flow-monitor)# cache timeout active 1800 例： Device(config-flow-monitor)# cache timeout inactive 200 例： Device(config-flow-monitor)# cache type normal	(任意) フローキャッシュパラメータを設定するように指定します。 • entries <i>number-of-entries</i> : フローキャッシュ内のフローエントリの最大数を 16 ~ 65536 の範囲で指定します。 (注) 標準のキャッシュタイプのみがサポートされます。
ステップ 7	end 例： Device(config)# end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバルコンフィギュレーションモードを終了できます。
ステップ 8	show flow monitor 例： Device# show flow monitor	すべてのフローモニタに関する情報を表示します。
ステップ 9	show flow monitor <i>flow-monitor-name</i> 例： Device# show flow monitor flow-monitor-1	指定した有線 AVC フロー モニタに関する情報を表示します。
ステップ 10	show flow monitor <i>flow-monitor-name</i> statistics 例： Device# show flow monitor flow-monitor-1 statistics	有線 AVC フロー モニタの統計情報を表示します。

	コマンドまたはアクション	目的
ステップ 11	clear flow monitor <i>flow-monitor-name</i> statistics 例 : <pre>Device# clear flow monitor flow-monitor-1 statistics</pre>	指定したフローモニタの統計情報をクリアします。 clear flow monitor flow-monitor-1 statistics を使用した後に show flow monitor flow-monitor-1 statistics コマンドを使用して、すべての統計情報がリセットされたことを確認します。
ステップ 12	show flow monitor <i>flow-monitor-name</i> cache format table 例 : <pre>Device# show flow monitor flow-monitor-1 cache format table</pre>	表形式でフローキャッシュの内容を表示します。
ステップ 13	show flow monitor <i>flow-monitor-name</i> cache format record 例 : <pre>Device# show flow monitor flow-monitor-1 cache format record</pre>	フローレコードと同様の形式でフローキャッシュの内容を表示します。
ステップ 14	show flow monitor <i>flow-monitor-name</i> cache format csv 例 : <pre>Device# show flow monitor flow-monitor-1 cache format csv</pre>	CSV形式でフローキャッシュの内容を表示します。

インターフェイスへのフロー モニタの関連付け

異なる事前定義済みレコードを持つ 2 つの異なる有線 AVC モニタをインターフェイスに同時に接続できます。

手順

	コマンドまたはアクション	目的
ステップ 1	configure terminal 例 : <pre>Device# configure terminal</pre>	グローバル コンフィギュレーション モードを開始します。
ステップ 2	interface <i>interface-id</i> 例 : <pre>Device(config)# interface Gigabitethernet 1/0/1</pre>	インターフェイスコンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 3	ip flow monitor <i>monitor-name</i> { input output } 例 : Device(config-if) # ip flow monitor flow-monitor-1 input	入力パケットと出力パケットの両方またはいずれか用のインターフェイスにフロー モニタを関連付けます。
ステップ 4	end 例 : Device(config) # end	特権 EXEC モードに戻ります。また、Ctrl+Z キーを押しても、グローバル コンフィギュレーション モードを終了できます。

NBAR2 カスタム アプリケーション

NBAR2 では、カスタム プロトコルを使用してカスタム アプリケーションを識別できます。カスタム プロトコルは、プロトコルとアプリケーションをサポートしますが、現在のところ、NBAR2 はサポートしていません。

すべての展開において、シスコが提供する NBAR2 プロトコルパックの対象外であるローカル アプリケーションおよび特定のアプリケーションがあります。ローカル アプリケーションは主に次のように分類されます。

- 組織への特定のアプリケーション
- 地域特有のアプリケーション

NBAR2 では、このようなローカル アプリケーションを手動でカスタマイズする方法を提供しています。グローバル コンフィギュレーション モードで **ip nbar custom myappname** コマンドを使用して、手動でアプリケーションをカスタマイズできます。カスタム アプリケーションは、組み込みプロトコルより優先されます。それぞれのカスタム プロトコルでは、ユーザは、レポート目的に使用できるセレクト ID を定義できます。

さまざまなタイプのアプリケーション カスタマイズがあります。

一般的なプロトコルのカスタマイズ

- HTTP
- SSL
- DNS

コンポジット：複数の基本的なプロトコルに基づくカスタマイズ：**server-name**

レイヤ 3/レイヤ 4 のカスタマイズ

- IPv4 アドレス
- DSCP 値
- TCP/UDP ポート

- フロー送信元または宛先の方向

バイト オフセット：ペイロードの特定のバイト値に基づくカスタマイズ

HTTP のカスタマイズ

HTTP のカスタマイズは、次の HTTP フィールドの組み合わせに基づいて実行できます。

- **cookie** : HTTP クッキー
- **host** : リソースを含む元のサーバのホスト名
- **method** : HTTP メソッド
- **referrer** : リソース リクエストの取得元のアドレス
- **url** : Uniform Resource Locator のパス
- **user-agent** : 要求を送信するエージェントによって使用されているソフトウェア
- **version** : HTTP バージョン
- **via** : HTTP 経由フィールド

HTTP のカスタマイズ

セレクト ID 10 が付いた HTTP ホスト「*mydomain.com」を使用する MYHTTP と呼ばれるカスタム アプリケーション。

```
Device# configure terminal
Device(config)# ip nbar custom MYHTTP http host *mydomain.com id 10
```

SSL のカスタマイズ

SSL サーバ名指定 (SNI) または共通名 (CN) から抽出した情報を使用して、SSL 暗号化トラフィックでカスタマイズを行うことができます。

SSL のカスタマイズ

セレクト ID 11 が付いた SSL 固有名「mydomain.com」を使用する MYSSL と呼ばれるカスタム アプリケーション。

```
Device# configure terminal
Device(config)# ip nbar custom MYSSL ssl unique-name *mydomain.com id 11
```

DNS のカスタマイズ

NBAR2 は、DNS 要求および応答トラフィックを確認し、アプリケーションへの DNS 応答に関連付けることができます。DNS 応答から戻された IP アドレスはキャッシュされ、その特定のアプリケーションに関連付けられているその後のパケット フローに使用されます。

ip nbar custom application-name dns domain-name id application-id コマンドは、DNS のカスタマイズに使用されます。既存のアプリケーションを拡張するには、**ip nbar custom application-name dns domain-name domain-name extends existing-application** コマンドを使用します。

DNS ベースのカスタマイズの詳細については、http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_nbar/configuration/xr-3s/asr1000/qos-nbar-xr-3s-asr-1000-book/nbar-custapp-dns-xr.html を参照してください。

DNS のカスタマイズ

セレクタ ID 12 が付いた DNS ドメイン名「mydomain.com」を使用する MYDNS と呼ばれるカスタム アプリケーション。

```
Device# configure terminal
Device(config)# ip nbar custom MYDNS dns domain-name *mydomain.com id 12
```

複合カスタマイズ

NBAR2 では、HTTP、SSL または DNS に現れるドメイン名に基づいてアプリケーションをカスタマイズする方法が提供されます。

複合カスタマイズ

セレクタ ID 13 が付いた HTTP、SSL または DNS ドメイン名「mydomain.com」を使用する MYDOMAIN と呼ばれるカスタム アプリケーション。

```
Device# configure terminal
Device(config)# ip nbar custom MYDOMAIN composite server-name *mydomain.com id 13
```

L3/L4 のカスタマイズ

レイヤ3/レイヤ4のカスタマイズは、パケットタプルに基づいており、フローの最初のパケットで常に一致します。

L3/L4 のカスタマイズ

IP アドレス 10.56.1.10 および 10.56.1.11、セレクタ ID 14 が付いた TCP および DSCP ef に一致する LAYER4CUSTOM と呼ばれるカスタム アプリケーション。

```
Device# configure terminal
Device(config)# ip nbar custom LAYER4CUSTOM transport tcp id 14
Device(config-custom)# ip address 10.56.1.10 10.56.1.11
Device(config-custom)# dscp ef
```

例：カスタム アプリケーションのモニタリング

カスタム アプリケーションのモニタリングのための **show** コマンド
show ip nbar protocol-id | inc Custom

```
Device# show ip nbar protocol-id | inc Custom
LAYER4CUSTOM          14          Custom
MYDNS                 12          Custom
MYDOMAIN              13          Custom
MYHTTP                10          Custom
MYSSL                 11          Custom
```

show ip nbar protocol-discovery protocol CUSTOM_APP

```
Device# show ip nbar protocol-id MYSSL
Protocol Name          id          type
-----
MYSSL                  11          Custom
```

NBAR2 ダイナミック ヒットレス プロトコル パックのアップグレード

プロトコルパックは、デバイスのシスコソフトウェアを置き換えることなく、デバイスの NBAR2 プロトコル サポートを更新するソフトウェア パッケージです。プロトコルパックには、NBAR2 によって正式にサポートされている、コンパイル済みでパック済みのアプリケーションに関する情報が含まれています。各アプリケーションについて、プロトコルパックには、アプリケーション署名とアプリケーション属性の情報が含まれています。各ソフトウェア リリースには、組み込みのプロトコルパックがバンドルされています。

プロトコルパックには次の特長があります。

- ロードが容易で高速。
- 高いバージョンのプロトコルパックにアップグレードしたり、低いバージョンのプロトコルパックに戻したりするのが容易。
- スイッチのリロードを必要としない。

NBAR2 プロトコルパックは、次の URL から Cisco Software Center でダウンロードできます：
<https://software.cisco.com/download/home>

NBAR2 プロトコルパックの前提条件

新しいプロトコルパックをロードする前に、すべてのスイッチ メンバー上でプロトコルパックをフラッシュにコピーする必要があります。

プロトコルパックをロードするには、[NBAR2 プロトコルパックのロード \(31 ページ\)](#) を参照してください。

NBAR2 プロトコルパックのロード

手順

	コマンドまたはアクション	目的
ステップ 1	enable 例： Device> enable	特権 EXEC モードを有効にします。 <ul style="list-style-type: none"> • パスワードを入力します（要求された場合）。

例 : NBAR2 プロトコルパックのロード

	コマンドまたはアクション	目的
ステップ 2	configure terminal 例 : Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 3	ip nbar protocol-pack protocol-pack [force] 例 : Device(config)# ip nbar protocol-pack flash:defProtoPack 例 : Device(config)# default ip nbar protocol-pack	プロトコルパックをロードします。 <ul style="list-style-type: none"> • 基本のプロトコルパックバージョンとは異なる、より低いバージョンのプロトコルパックを指定し、ロードするには、force キーワードを使用します。これにより、スイッチの現在のプロトコルパックでサポートされていない設定も削除されます。 組み込みのプロトコルパックに戻るには、次のコマンドを使用します。
ステップ 4	exit 例 : Device(config)# exit	特権 EXEC モードに戻ります。
ステップ 5	show ip nbar protocol-pack {protocol-pack active} [detail] 例 : Device# show ip nbar protocol-pack active	プロトコルパック情報を表示します。 <ul style="list-style-type: none"> • このコマンドを使用して、ロードされたプロトコルパックのバージョン、パブリッシャ、その他の詳細を確認します。 • 指定されたプロトコルパックの情報を表示するには、<i>protocol-pack</i> 引数を使用します。 • アクティブなプロトコルパックの情報を表示するには、active キーワードを使用します。 • 詳細なプロトコルパックの情報を表示するには、detail キーワードを使用します。

例 : NBAR2 プロトコルパックのロード

次の例に、新しいプロトコルパックをロードする方法を示します。

```
Device> enable
Device# configure terminal
```



```
Device(config)# ip nbar protocol-pack flash:newDefProtoPack
Device(config)# exit
```

次の例に、**force** キーワードを使用して下位バージョンのプロトコルパックをロードする方法を示します。

```
Device> enable
Device# configure terminal
Device(config)# ip nbar protocol-pack flash:OldDefProtoPack force
Device(config)# exit
```

次の例に、組み込みのプロトコルパックに戻す方法を示します。

```
Device> enable
Device# configure terminal
Device(config)# default ip nbar protocol-pack
Device(config)# exit
```

Application Visibility and Control のモニタリング

このセクションでは、アプリケーションの可視性に関する新しいコマンドについて説明します。

次のコマンドは、スイッチおよびアクセスポートのアプリケーションの可視性をモニタするために使用できます。

表 1: スイッチのアプリケーションの可視性モニタリングコマンド

コマンド	目的
<pre>show ip nbar protocol-discovery [interface interface-type interface-number] [stats{byte-count bit-rate packet-count max-bit-rate}] [protocol protocol-name top-n number]</pre>	<p>NBAR Protocol Discovery 機能によって収集された統計情報を表示します。</p> <ul style="list-style-type: none"> (任意) 表示される統計情報を最適化するには、キーワードおよび引数を入力します。キーワードのそれぞれの詳細については、『Cisco IOS Quality of Service Solutions Command Reference』の show ip nbar protocol-discovery コマンドを参照してください。
<pre>show policy-map interface interface-type interface-number</pre>	<p>インターフェイスに適用したポリシー マップについての情報を表示します。</p>

例 : Application Visibility and Control の設定

次に、**match protocol** でアプリケーション名のフィルタを適用してクラス マップを作成する例を示します。

```
Device# configure terminal
Device(config)# class-map match-any NBAR-VOICE
```

```
Device(config-cmap)# match protocol ms-lync-audio
Device(config-cmap)#end
```

次に、ポリシー マップを作成し、出力 QoS の既存のクラス マップを定義する例を示します。

```
Device# configure terminal
Device(config)# policy-map test-avc-up
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 150000
Device(config-pmap-c)# set dscp 12
Device(config-pmap-c)#end
```

次に、ポリシー マップを作成し、入力 QoS の既存のクラス マップを定義する例を示します。

```
Device# configure terminal
Device(config)# policy-map test-avc-down
Device(config-pmap)# class cat-browsing
Device(config-pmap-c)# police 200000
Device(config-pmap-c)# set dscp 10
Device(config-pmap-c)#end
```

次に、ポリシー マップをスイッチ ポートに適用する例を示します。

```
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# switchport mode access
Device(config-if)# switchport access vlan 20
Device(config-if)# service-policy input POLICING_IN
Device(config-if)#end
```

次に、NBAR 属性に基づいてクラスマップを作成する例を示します。

```
Device# configure terminal
Device(config)# class-map match-all rel-relevant
Device(config-cmap)# match protocol attribute business-relevance business-relevant

Device(config)# class-map match-all rel-irrelevant
Device(config-cmap)# match protocol attribute business-relevance business-irrelevant

Device(config)# class-map match-all rel-default
Device(config-cmap)# match protocol attribute business-relevance default

Device(config)# class-map match-all class--ops-admin-and-rel
Device(config-cmap)# match protocol attribute traffic-class ops-admin-mgmt
Device(config-cmap)# match protocol attribute business-relevance business-relevant
```

次に、NBAR 属性に基づくクラスマップに基づいてポリシーマップを作成する例を示します。

```
Device# configure terminal
Device(config)# policy-map attrib--rel-types
Device(config-pmap)# class rel-relevant
Device(config-pmap-c)# set dscp ef
Device(config-pmap-c)# class rel-irrelevant
Device(config-pmap-c)# set dscp af11
Device(config-pmap-c)# class rel-default
Device(config-pmap-c)# set dscp default

Device(config)# policy-map attrib--ops-admin-and-rel
Device(config-pmap)# class class--ops-admin-and-rel
Device(config-pmap-c)# set dscp cs5
```

次に、NBAR 属性に基づくポリシーマップを有線ポートに適用する例を示します。

```
Device# configure terminal
Device(config)# interface GigabitEthernet1/0/2
Device(config-if)# service-policy input attrib--rel-types
```

show コマンドによる設定の表示

show ip nbar protocol-discovery

インターフェイスごとのプロトコル検出統計情報のレポートを表示します。

次に、インターフェイスごとの統計情報の出力例を示します。

```
Device# show ip nbar protocol-discovery int GigabitEthernet1/0/1

GigabitEthernet1/0/1
Last clearing of "show ip nbar protocol-discovery" counters 00:03:16

Output
-----
Protocol          Packet Count
Packet Count      Byte Count
Byte Count        30sec Bit Rate (bps)
30sec Bit Rate (bps) 30sec Max Bit Rate (bps)
30sec Max Bit Rate (bps)
-----
ms-lync           60580
55911
28774864          31174777
3613000
93000             3613000
3437000
Total             60580
55911
28774864          31174777
3613000
93000             3613000
3437000
```

show policy-map interface

すべてのインターフェイス上の QoS 統計情報および設定済みのポリシーマップを表示します。

次に、すべてのインターフェイスに設定されたポリシーマップの出力例を示します。

```

Device# show policy-map int

GigabitEthernet1/0/1
  Service-policy input: MARKING-IN

  Class-map: NBAR-VOICE (match-any)
    718 packets
  Match: protocol ms-lync-audio
    0 packets, 0 bytes
    30 second rate 0 bps
  QoS Set
    dscp ef

  Class-map: NBAR-MM_CONFERENCING (match-any)
    6451 packets
  Match: protocol ms-lync
    0 packets, 0 bytes
    30 second rate 0 bps
  Match: protocol ms-lync-video
    0 packets, 0 bytes
    30 second rate 0 bps
  QoS Set
    dscp af41

  Class-map: class-default (match-any)
    34 packets
  Match: any

```

show コマンドによる属性ベースの QoS 設定の表示

show policy-map interface

すべてのインターフェイス上の属性ベースの QoS 統計情報および設定済みのポリシーマップを表示します。

次に、すべてのインターフェイスに設定されたポリシーマップの出力例を示します。

```

Device# show policy-map interface gigabitEthernet 1/0/2
GigabitEthernet1/0/2

  Service-policy input: attrib--rel-types

  Class-map: rel-relevant (match-all)
    20 packets
  Match: protocol attribute business-relevance business-relevant
  QoS Set
    dscp ef

  Class-map: rel-irrelevant (match-all)
    0 packets
  Match: protocol attribute business-relevance business-irrelevant

  QoS Set

```

```
dscp af11

Class-map: rel-default (match-all)
  14 packets
  Match: protocol attribute business-relevance default
  QoS Set
  dscp default

Class-map: class-default (match-any)
  0 packets
  Match: any
```

show ip nbar protocol-attribute

NBAR で使用されるすべてのプロトコル属性を表示します。

次に、一部の属性の出力例を示します。

```
Device# show ip nbar protocol-attribute cisco-jabber-im
  Protocol Name : cisco-jabber-im
    encrypted : encrypted-yes
    tunnel : tunnel-no
    category : voice-and-video
    sub-category : enterprise-media-conferencing
  application-group : cisco-jabber-group
  p2p-technology : p2p-tech-no
  traffic-class : transactional-data
  business-relevance : business-relevant
  application-set : collaboration-apps

Device# show ip nbar protocol-attribute google-services
  Protocol Name : google-services
    encrypted : encrypted-yes
    tunnel : tunnel-no
    category : other
    sub-category : other
  application-group : google-group
  p2p-technology : p2p-tech-yes
  traffic-class : transactional-data
  business-relevance : default
  application-set : general-browsing

Device# show ip nbar protocol-attribute dns
  Protocol Name : google-services
    encrypted : encrypted-yes
    tunnel : tunnel-no
    category : other
    sub-category : other
  application-group : google-group
  p2p-technology : p2p-tech-yes
  traffic-class : transactional-data
  business-relevance : default
  application-set : general-browsing
```

```
Device# show ip nbar protocol-attribute unknown
      Protocol Name : unknown
      encrypted    : encrypted-no
      tunnel       : tunnel-no
      category     : other
      sub-category : other
      application-group : other
      p2p-technology : p2p-tech-no
      traffic-class : bulk-data
      business-relevance : default
      application-set : general-misc
```

show コマンドによるフロー モニタ設定の表示

show flow monitor wdavc

指定した有線 AVC フロー モニタに関する情報を表示します。

```
Device # show flow monitor wdavc

Flow Monitor wdavc:
  Description:      User defined
  Flow Record:     wdavc
  Flow Exporter:   wdavc-exp (inactive)
  Cache:
    Type:          normal (Platform cache)
    Status:        not allocated
    Size:          12000 entries
    Inactive Timeout: 15 secs
    Active Timeout: 1800 secs
```

show flow monitor wdavc statistics

有線 AVC フロー モニタの統計情報を表示します。

```
Device# show flow monitor wdavc statistics
Cache type:          Normal (Platform cache)
Cache size:          12000
Current entries:     13

Flows added:        26
Flows aged:         13
  - Active timeout   ( 1800 secs) 1
  - Inactive timeout (   15 secs) 12
```

clear flow monitor wdavc statistics

指定したフロー モニタの統計情報をクリアします。**clear flow monitor wdavc statistics** を使用した後に **show flow monitor wdavc statistics** コマンドを使用して、すべての統計情報がリセットされたことを確認します。以下に、フローモニタ統計情報をクリアした後の **show flow monitor wdavc statistics** コマンドのサンプル出力を示します。

```
Device# show flow monitor wdavc statistics
Cache type:          Normal (Platform cache)
Cache size:          12000
Current entries:     0
```

```

Flows added:          0
Flows aged:           0

```

show コマンドによるキャッシュの内容の表示

show flow monitor wdvac cache format table

表形式でフロー キャッシュの内容を表示します。

```

Device# show flow monitor wdvac cache format table
Cache type:          Normal (Platform cache)
Cache size:          12000
Current entries:     13

Flows added:         26
Flows aged:          13
- Active timeout    ( 1800 secs)  1
- Inactive timeout  (   15 secs)  12

CONN IPV4 INITIATOR ADDR  CONN IPV4 RESPONDER ADDR  CONN RESPONDER PORT
FLOW OBSPOINT ID  IP VERSION  IP PROT  APP NAME
flow dirn .....
-----
-----
64.103.125.147          144.254.71.184
53          4294967305          4          17  port dns
  Input .....
64.103.121.103          10.1.1.2
67          4294967305          4          17  layer7 dhcp
  Input ....contd.....
64.103.125.3           64.103.125.97
68          4294967305          4          17  layer7 dhcp
  Input .....
10.0.2.6                157.55.40.149          443
          4294967305          4          6   layer7 ms-lync
  Input .....
64.103.126.28          66.163.36.139          443
          4294967305          4          6   layer7 cisco-jabber-im
  Input ....contd.....
64.103.125.2           64.103.125.29
68          4294967305          4          17  layer7 dhcp
  Input .....
64.103.125.97          64.103.101.181
67          4294967305          4          17  layer7 dhcp
  Input .....
192.168.100.6          10.10.20.1          5060
          4294967305          4          17  layer7 cisco-jabber-control
  Input ....contd.....
64.103.125.3           64.103.125.29
68          4294967305          4          17  layer7 dhcp
  Input .....

```

```

10.80.101.18          10.80.101.6          5060
      4294967305          4          6 layer7 cisco-collab-control
Input .....
10.1.11.4            66.102.11.99
80      4294967305          4          6 layer7 google-services
      Input ....contd.....
64.103.125.2        64.103.125.97
68      4294967305          4          17 layer7 dhcp
      Input .....
64.103.125.29       64.103.101.181
67      4294967305          4          17 layer7 dhcp
      Input .....

```

show flow monitor wdavc cache format record

フローレコードと同様の形式でフローキャッシュの内容を表示します。

```

Device# show flow monitor wdavc cache format record
Cache type: Normal (Platform cache)
Cache size: 12000
Current entries: 13

Flows added: 26
Flows aged: 13
- Active timeout ( 1800 secs) 1
- Inactive timeout ( 15 secs) 12

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.147
CONNECTION IPV4 RESPONDER ADDRESS: 144.254.71.184
CONNECTION RESPONDER PORT: 53
FLOW OBSPOINT ID: 4294967305
IP VERSION: 4
IP PROTOCOL: 17
APPLICATION NAME: port dns
flow direction: Input
timestamp abs first: 08:55:46.917
timestamp abs last: 08:55:46.917
connection initiator: Initiator
connection count new: 2
connection serverpackets counter: 1
connection client packets counter: 1
connection server network bytes counter: 190
connection client network bytes counter: 106

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.121.103
CONNECTION IPV4 RESPONDER ADDRESS: 10.1.1.2
CONNECTION RESPONDER PORT: 67
FLOW OBSPOINT ID: 4294967305
IP VERSION: 4
IP PROTOCOL: 17
APPLICATION NAME: layer7 dhcp
flow direction: Input
timestamp abs first: 08:55:47.917

```



```
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 1
connection server network bytes counter: 0
connection client network bytes counter: 350

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.3
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.125.97
CONNECTION RESPONDER PORT:        68
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:53.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 4
connection server network bytes counter: 0
connection client network bytes counter: 1412

CONNECTION IPV4 INITIATOR ADDRESS: 10.0.2.6
CONNECTION IPV4 RESPONDER ADDRESS: 157.55.40.149
CONNECTION RESPONDER PORT:        443
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       6
APPLICATION NAME:                   layer7 ms-lync
flow direction:                     Input
timestamp abs first:                08:55:46.917
timestamp abs last:                08:55:46.917
connection initiator:              Initiator
connection count new:              2
connection server packets counter: 10
connection client packets counter: 14
connection server network bytes counter: 6490
connection client network bytes counter: 1639

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.126.28
CONNECTION IPV4 RESPONDER ADDRESS: 66.163.36.139
CONNECTION RESPONDER PORT:        443
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       6
APPLICATION NAME:                   layer7 cisco-jabber-im
flow direction:                     Input
timestamp abs first:                08:55:46.917
```

```
timestamp abs last:                08:55:46.917
connection initiator:              Initiator
connection count new:              2
connection server packets counter: 12
connection client packets counter: 10
connection server network bytes counter: 5871
connection client network bytes counter: 2088

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.2
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.125.29
CONNECTION RESPONDER PORT:        68
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 2
connection server network bytes counter: 0
connection client network bytes counter: 712

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.97
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.101.181
CONNECTION RESPONDER PORT:        67
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 1
connection server network bytes counter: 0
connection client network bytes counter: 350

CONNECTION IPV4 INITIATOR ADDRESS: 192.168.100.6
CONNECTION IPV4 RESPONDER ADDRESS: 10.10.20.1
CONNECTION RESPONDER PORT:        5060
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 cisco-jabber-control
flow direction:                     Input
timestamp abs first:                08:55:46.917
```

```
timestamp abs last:                08:55:46.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 2
connection server network bytes counter: 0
connection client network bytes counter: 2046

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.3
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.125.29
CONNECTION RESPONDER PORT:        68
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 2
connection server network bytes counter: 0
connection client network bytes counter: 712

CONNECTION IPV4 INITIATOR ADDRESS: 10.80.101.18
CONNECTION IPV4 RESPONDER ADDRESS: 10.80.101.6
CONNECTION RESPONDER PORT:        5060
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       6
APPLICATION NAME:                   layer7 cisco-collab-control
flow direction:                     Input
timestamp abs first:                08:55:46.917
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              2
connection server packets counter: 23
connection client packets counter: 27
connection server network bytes counter: 12752
connection client network bytes counter: 8773

CONNECTION IPV4 INITIATOR ADDRESS: 10.1.11.4
CONNECTION IPV4 RESPONDER ADDRESS: 66.102.11.99
CONNECTION RESPONDER PORT:        80
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       6
APPLICATION NAME:                   layer7 google-services
flow direction:                     Input
timestamp abs first:                08:55:46.917
```

```

timestamp abs last:                08:55:46.917
connection initiator:              Initiator
connection count new:              2
connection server packets counter: 3
connection client packets counter: 5
connection server network bytes counter: 1733
connection client network bytes counter: 663

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.2
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.125.97
CONNECTION RESPONDER PORT:        68
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:53.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 4
connection server network bytes counter: 0
connection client network bytes counter: 1412

CONNECTION IPV4 INITIATOR ADDRESS: 64.103.125.29
CONNECTION IPV4 RESPONDER ADDRESS: 64.103.101.181
CONNECTION RESPONDER PORT:        67
FLOW OBSPOINT ID:                 4294967305
IP VERSION:                        4
IP PROTOCOL:                       17
APPLICATION NAME:                   layer7 dhcp
flow direction:                     Input
timestamp abs first:                08:55:47.917
timestamp abs last:                08:55:47.917
connection initiator:              Initiator
connection count new:              1
connection server packets counter: 0
connection client packets counter: 1
connection server network bytes counter: 0
connection client network bytes counter: 350

```

show flow monitor wdavc cache format csv

CSV 形式でフロー キャッシュの内容を表示します。

```

Device# show flow monitor wdavc cache format csv
Cache type:                Normal (Platform cache)
Cache size:                 12000
Current entries:            13

Flows added:                26
Flows aged:                 13

```

```
- Active timeout      ( 1800 secs)      1
- Inactive timeout   (   15 secs)     12
```

```
CONN IPV4 INITIATOR ADDR,CONN IPV4 RESPONDER ADDR,CONN RESPONDER
PORT,FLOW OBSPOINT ID,IP VERSION,IP
PROT,APP NAME,flow dirn,time abs first,time abs last,conn initiator,conn
count new,conn server packets
cnt,conn client packets cnt,conn server network bytes cnt,conn client
network bytes cnt
64.103.125.147,144.254.71.184,53,4294967305,4,17,port
dns,Input,08:55:46.917,08:55:46.917,Initiator,2,1,1,190,106
64.103.121.103,10.1.1.2,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350
64.103.125.3,64.103.125.97,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:53.917,Initiator,1,0,4,0,1412
10.0.2.6,157.55.40.149,443,4294967305,4,6,layer7 ms-
lync,Input,08:55:46.917,08:55:46.917,Initiator,2,10,14,6490,1639
64.103.126.28,66.163.36.139,443,4294967305,4,6,layer7 cisco-jabber-
im,Input,08:55:46.917,08:55:46.917,Initiator,2,12,10,5871,2088
64.103.125.2,64.103.125.29,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,2,0,712
64.103.125.97,64.103.101.181,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350
192.168.100.6,10.10.20.1,5060,4294967305,4,17,layer7 cisco-jabber-
control,Input,08:55:46.917,08:55:46.917,Initiator,1,0,2,0,2046
64.103.125.3,64.103.125.29,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,2,0,712
10.80.101.18,10.80.101.6,5060,4294967305,4,6,layer7 cisco-collab-
control,Input,08:55:46.917,08:55:47.917,Initiator,2,23,27,12752,8773
10.1.11.4,66.102.11.99,80,4294967305,4,6,layer7 google-
services,Input,08:55:46.917,08:55:46.917,Initiator,2,3,5,1733,663
64.103.125.2,64.103.125.97,68,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:53.917,Initiator,1,0,4,0,1412
64.103.125.29,64.103.101.181,67,4294967305,4,17,layer7
dhcp,Input,08:55:47.917,08:55:47.917,Initiator,1,0,1,0,350
```

基本的なトラブルシューティング：質問と回答

以下に、有線 Application Visibility and Control のトラブルシューティングに関する基本的な質問と回答を示します。

1. 質問：IPv6 トラフィックが分類されていません。
回答：現在は IPv4 トラフィックのみがサポートされています。
2. 質問：マルチキャスト トラフィックが分類されていません。
回答：現在はユニキャスト トラフィックのみがサポートされています。
3. 質問：ping を送信したときに、分類されているかを確認できません。

回答：TCP/UDP プロトコルのみがサポートされています。

4. 質問：SVI に NBAR を接続できないのはなぜですか。

回答：NBAR は物理インターフェイスでのみサポートされています。

5. 質問：ほとんどのトラフィックが CAPWAP トラフィックになっているのですが、なぜですか。

回答：ワイヤレス アクセス ポートに接続されていないアクセス ポートで NBAR が有効になっていることを確認してください。AP から着信するすべてのトラフィックは capwap として分類されます。この場合、実際の分類は AP または WLC で行われます。

6. 質問：プロトコル検出で、トラフィックが片側でしか確認できません。さらに、多くの未知のトラフィックがあります。

回答：これは通常、NBAR が非対称トラフィックを確認していることを示します。片側のトラフィックは1つのスイッチメンバーに分類され、もう一方は別のメンバーに分類されます。トラフィックの両側が確認されるアクセスポートにのみ NBAR を接続することを推奨します。複数のアップリンクがある場合は、この問題のためそれらに NBAR を接続することはできません。ポートチャネルの一部であるインターフェイスに NBAR を設定した場合にも同様の問題が発生します。

7. 質問：プロトコル検出で、すべてのアプリケーションの集約ビューが表示されます。時間経過に伴うトラフィック分布を確認するにはどうしたらいいですか。

回答：WebUI を使用して、過去 48 時間の経時的なトラフィックを表示できます。

8. 質問：`match protocol protocol-name` コマンドを使用してキューベースのイーグレスポリシーを設定できません。

回答：NBAR2 ベースの分類子が含まれるポリシーでは、**shape** および **set DSCP** のみがサポートされています。一般的な方法としては、入力で DSCP を設定し、DSCP に基づいて出力でシェーピングを実行します。

9. 質問：インターフェイスに接続している NBAR2 はありませんが、NBAR2 がいまだにアクティブになっています。

回答：`match protocol protocol-name` を含むクラスマップがあると、NBAR はスタックでグローバルにアクティブになりますが、トラフィックは NBAR 分類の対象にはなりません。これは予期された動作であり、リソースを消費しません。

10. 質問：デフォルトの QOS キューの下にトラフィックがあります。どうしてですか。

回答：新しい各フローでは、フローを分類してハードウェアに結果をインストールするためにいくつかの packets が使われます。この間に、分類は「不明」となり、トラフィックはデフォルトキューに入ります。

Application Visibility and Control に関する追加情報

関連資料

関連項目	マニュアル タイトル
この章で使用するコマンドの完全な構文および使用方法の詳細。	<i>Command Reference (Catalyst 9200 Series Switches)</i>

有線ネットワークでの Application Visibility and Control の機能履歴

次の表に、このモジュールで説明する機能のリリースおよび関連情報を示します。

これらの機能は、特に明記されていない限り、導入されたリリース以降のすべてのリリースで使用できます。

リリース	機能	機能情報
Cisco IOS XE Fuji 16.8.1a	有線アプリケーションの表示およびコントロール（有線 AVC）属性ベース QoS（EasyQoS）	特定のプロトコルではなく、Network-Based Application Recognition（NBAR）属性に基づいて QoS クラスとポリシーを定義できるようになりましたが、いくつかの制限があります。サポートされる NBAR 属性は、business-relevance および traffic-class のみです。
Cisco IOS XE Gibraltar 16.11.1	有線ネットワークでの Application Visibility and Control	AVC は、アプリケーションへの適応力やアプリケーションへのインテリジェンス性に基づいて、厳密なパケットおよび接続からブランチおよびキャンパスソリューションを発展させるためのシスコの取り組みの重要な部分です。
Cisco IOS XE Gibraltar 16.12.1	DNS フローレコード	DNS フローレコードのサポートが導入されました。DNS フローレコードは、フローレコードを定義するための collect フィールドとして DNS ドメイン名を使用します。

Cisco Feature Navigator を使用すると、プラットフォームおよびソフトウェアイメージのサポート情報を検索できます。Cisco Feature Navigator には、<http://www.cisco.com/go/cfn> からアクセスします。

翻訳について

このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。