MIB の使用方法

この付録では、Cisco 7304 ルータ上でタスクを実行する方法を説明します。この付録の構成は、次のとおりです。

ルータ上の物理エンティティの管理

ここでは、ルータ上の物理エンティティ（コンポーネント）を管理する際の SNMP の使用方法を説明します。

目的および利点

Cisco 7304 の SNMP に実装されている物理エンティティの管理機能は、次のとおりです。

- Field Replaceable Unit（FRU）のステータスを監視および設定します。
- 物理ポートに関する情報をインターフェイス マッピングに提供します。
- シャーシ コンポーネントに関するファームウェアおよびソフトウェアの情報を提供します。

entPhysicalTable は、システムに含まれている物理エンティティを列挙します。さらに、各物理エンティティをベンダー タイプとクラスごとに分類します。

（注）　この付録に示されている出力および値の例は、MIB の使用時に表示される例に過ぎません。

物理エンティティの管理に使用される MIB

Cisco 7304 上の物理エンティティを管理するときに使用される MIB は、次のとおりです。

- CISCO-ENTITY-ASSET-MIB：ENTITY-MIB の entPhysicalTable に示されている物理エンティティに関する資産トラッキング情報（ID PROM コンテンツ）が含まれています。この MIB は、物理エンティティに関するデバイス固有の情報（発注時に使用する部品番号、シリアル番号、製造番号、ハードウェア情報、ソフトウェア情報、ファームウェア情報など）を提供します。
- CISCO-ENTITY-FRU-CONTROL-MIB：Field Replaceable Unit（FRU）の管理ステータスと動作ステータスを監視し、設定する際に使用されるオブジェクトが含まれています。FRU とは電源装置やラインカードなどで、ENTITY-MIB の entPhysicalTable に示されています。
- CISCO-ENTITY-VENDORTYPE-OID-MIB：entPhysicalTable にあるルータ上のすべての物理エンティティに関するオブジェクト ID（OID）が含まれています。
- CISCO-ENVMON-MIB：環境センサーのステータス情報が含まれています。たとえば、この MIB はシャーシのコア温度や吸気口の温度をレポートします。
- ENTITY-MIB：ルータ上の物理エンティティの管理に関する情報が含まれています。また、この MIB は、システムに含まれている物理エンティティを包含ツリーと呼ばれるツリーに編成します。このツリーは、物理エンティティそれぞれの相互関係を示します。ENTITY-MIB には、次のテーブルが含まれています。
  - entPhysicalTable は、ルータ内の各物理コンポーネント（エンティティ）を示します。このテーブルには、上位エンティティ（シャーシ）のエントリ、およびシャーシ内のエンティティごとのエントリが含まれています。各エントリは、そのエンティティに関する情報（名前、タイプ、ベンダー、説明など）を提供し、そのエンティティがシャーシ エンティティの階層構造にどう適合するかを示します。各エンティティは、この MIB およびその他の MIB 上のエンティティ情報にアクセスするために使用される一意のインデックス（*entPhysicalIndex*）によって識別されます。

entPhysicalTable には、次の 2 つの管理対象オブジェクトが含まれています。これらのオブジェクトは物理エンティティとその親との関係を示します。

- entPhysicalContainedIn：このオブジェクトは、この物理エンティティの親（つまり、この物理エンティティを含む物理エンティティ）を表す物理エンティティの entPhysicalIndex を指定します。たとえば、通常、スロット（シャーシ内の）を表す物理エンティティはに、ラインカードを表す物理エンティティが含まれます。したがって、スロットはラインカードの親となります。
- entPhysicalParentRelPos：このオブジェクトは、この物理エンティティとその親の相対関係を表す整数値を指定します。たとえば、スロットを表す物理エンティティには、物理エンティティが表す物理スロットのスロット番号に相当する entPhysicalParentRelPos が含まれることになります。
  - entPhysicalContainsTable は、システムに含まれている各物理エンティティの子を示します。このテーブルは、物理エンティティ間の親子関係のもう 1 つのビューを管理対象クライアントに提供します。

インベントリ管理の実行

Cisco 7304 ルータ内の特定の物理エンティティに関する情報を入手するには、ENTITY-MIB の entPhysicalTable を調べます。entPhysicalDescr は、エンティティの説明をテキストで提供します。 エンティティの説明に該当する entPhysicalIndex を使用して、他のパラメータを取得できます。

entPhysicalTable のエントリに関する注意

ENTITY-MIB 内の entPhysicalTable のエントリを調べるときは、次の点に注意してください。

- entPhysicalIndex：シャーシ内の各エンティティを一意に識別します。 このインデックスは、他の MIB のエンティティ情報にアクセスする際にも使用されます。
- entPhysicalContainedIn：コンポーネントの親エンティティの entPhysicalIndex を表します。
- entPhysicalParentRelPos：同じ entPhysicalContainedIn 値をもつ同タイプのエンティティ（シャーシ スロット、ラインカード ポートなど）間の相対的な位置を示します。

（注）　コンテナを使用できるのは、1　つまたは複数のリムーバブル物理エンティティを物理エンティティのクラスに含めることができる場合です。たとえば、シャーシ内の各（空またはフル）スロットをコンテナとしてモデル化できます。すべてのリムーバブル物理エンティティは、コンテナ エンティティ（フィールドで交換可能なモジュール、ファン、電源装置など）内にモデル化します。

ここの例では、entPhysicalTable 内のエントリがエンティティ情報を提供する方法を示しています。各エンティティを調べる前に、次の出力で MIB ウォーク（walk）情報を表示する方法を示します（これは出力の一部に過ぎません）。

```
entPhysicalDescr.1 = 7300 Chassis
entPhysicalDescr.2 = Chassis Slot
entPhysicalDescr.3 = Chassis Slot
entPhysicalDescr.4 = Chassis Slot
entPhysicalDescr.5 = Chassis Slot
entPhysicalDescr.6 = Chassis Slot
entPhysicalDescr.7 = Chassis Slot
entPhysicalDescr.8 = Power Supply Slot
entPhysicalDescr.9 = Power Supply Slot
entPhysicalDescr.10 = Network Service Engine 100 CPU Card
entPhysicalDescr.11 = Network Service Engine 100 Daughter Card
entPhysicalDescr.12 = Mistral EOBC
entPhysicalDescr.13 = GBIC Port Container
entPhysicalDescr.14 = 1000BaseSX
entPhysicalDescr.15 = Pinnacle GE
entPhysicalVendorType.1 = cevChassis7304
entPhysicalVendorType.2 = cevContainerSlot
entPhysicalVendorType.3 = cevContainerSlot
entPhysicalVendorType.4 = cevContainerSlot
entPhysicalVendorType.5 = cevContainerSlot
entPhysicalVendorType.3 = cevContainerSlot
entPhysicalVendorType.7 = cevContainerSlot
entPhysicalVendorType.8 = cevContainerC7304PowerSupplyBay
entPhysicalVendorType.9 = cevContainerC7304PowerSupplyBay
entPhysicalVendorType.10 = cevCpuC7300Nse100
entPhysicalVendorType.11 = cevC7300Nse100Db
entPhysicalVendorType.12 = cevPortFEIP
entPhysicalVendorType.13 = cevContainerGbic
entPhysicalVendorType.14 = cevMGBIC1000BaseSX
entPhysicalVendorType.15 = cevPortGe
entPhysicalContainedIn.1 = 0
entPhysicalContainedIn.2 = 1
entPhysicalContainedIn.3 = 1
entPhysicalContainedIn.4 = 1
entPhysicalContainedIn.5 = 1
entPhysicalContainedIn.6 = 1
entPhysicalContainedIn.7 = 1
entPhysicalContainedIn.8 = 1
entPhysicalContainedIn.9 = 1
entPhysicalContainedIn.10 = 2
entPhysicalContainedIn.11 = 10
entPhysicalContainedIn.12 = 10
entPhysicalContainedIn.13 = 10
entPhysicalContainedIn.14 = 13
entPhysicalContainedIn.15 = 14
.
.
.
```

ENTITY-MIB 内の entPhysicalTable エントリの例

ここの出力例では、情報を entPhysicalTable に保存する方法を示しています。entPhysicalTable のエントリを調べて、Cisco 7304 ルータの構成を確認できます。たとえば、entPhysicalIndex = 1 はシャーシタイプに相当し、それ以外の情報（ベンダー タイプ、物理クラスなど）は、このテーブルを調べることで入手できます。

（注）　entPhysicalEntry.*entPhysicalIndex* は、シャーシ内の各エンティティを一意に識別します。このインデックスを使用すると、他の MIB テーブル内のエンティティに関する情報にアクセスできます。

entPhysicalTEntry.1

```
entPhysicalDescr.1 = 7300 Chassis
entPhysicalVendorType.1 = cevChassis7304
entPhysicalContainedIn.1 = 0
entPhysicalClass.1 = chassis(3)
entPhysicalParentRelPos.1 = -1
entPhysicalName.1 = 7300 Chassis
entPhysicalHardwareRev.1 = E
entPhysicalFirmwareRev.1 =
entPhysicalSoftwareRev.1 =
entPhysicalSerialNum.1 = SCA070400D8
entPhysicalMfgName.1 = Cisco Systems Inc
entPhysicalModelName.1 = 73-5916-02
entPhysicalAlias.1 =
entPhysicalAssetID.1 =
entPhysicalIsFRU.1 = true(1)
.
.
.
```

表 A-1 では、entPhysicalEntry の出力例から入手できる entPhysicalEntry.1 および entPhysicalEntry.11 の構成情報を示しています。

表 A-1　entPhysicalEntry の構成情報

entPhysicalTEntries	entPhysicalTEntry の説明
entPhysicalIndex = 1	シャーシ タイプを識別します。
entPhysicalVendorType = cevChassis7304	物理エンティティのベンダー固有のハードウェア タイプを示します。 エージェントは、このオブジェクトを、特定の機器タイプを詳細に示す企業固有の登録 ID に設定します。entPhysicalClass の関連インスタンスを使用して、ハードウェア デバイスの汎用タイプ（自律タイプ）を示します。 ベンダー固有の登録 ID がこの物理エンティティにない場合、またはその値が不明である場合、値 00 が戻ります。
entPhysicalContainedIn = 0	0 ～ 2147483647 の整数。この物理エンティティを含む物理エンティティに対する entPhysicalIndex の値です。 コンテナを使用できるのは、1 つまたは複数のリムーバブル物理エンティティ（タイプが異なる場合もある）を物理エンティティのクラスに含めることができる場合です。たとえば、シャーシ内の各（空またはフル）スロットをコンテナとしてモデル化できます。 （注）　ゼロの値は、この物理エンティティが他のどの物理エンティティにも含まれていないことを示します。値がゼロということは、この物理エンティティがシャーシであることを示します。

表 A-1　entPhysicalEntry の構成情報 （続き）

entPhysicalTEntries	entPhysicalTEntry の説明
entPhysicalClass = chassis(3)	物理エンティティの汎用ハードウェア タイプ（シャーシ、モジュールなど）を示します。 エージェントは、このオブジェクトを、物理エンティティの一般クラス（複数のクラスがある場合は、プライマリクラス）を最も正確に示す標準列挙値に設定します。 適切な標準登録 ID がこの物理エンティティにない場合、値 other(1) が戻ります。値がこのエージェントで不明な場合は、値 unknown(2) が戻ります。
entPhysicalParentRelPos = -1	-1 ～ 2147483647 の整数。 子のすべての兄弟コンポーネントの中でその子コンポーネントの相対的な位置を示します。兄弟コンポーネントは、entPhysicalContainedIn オブジェクトおよび entPhysicalClass オブジェクトそれぞれの同じインスタンス値を共有する entPhysicalEntries として定義されます。 このオブジェクトは、特定の親のすべての兄弟コンポーネントの相対的な順序を示します（兄弟それぞれのエントリの entPhysicalContainedIn インスタンスによって識別されます）。この値は、できるだけ物理コンポーネントの外部ラベルと一致させるようにしてください。
entPhysicalName = 7300 Chassis	物理エンティティのテキスト名。このオブジェクトの値は、ローカル デバイスとして割り当てられたコンポーネント名でなければならず、デバイス コンソールで入力されるコマンドで使用できなければなりません。この値は、「console」などのテキスト名、または 1 などの単純なコンポーネント番号（ポートやモジュールの番号など）のいずれかとなります。どちらの値になるかは、デバイスの物理コンポーネントの命名構文によります。 ローカル名がない、またはその他の理由でこのオブジェクトが適用できない場合は、このオブジェクトに長さゼロの文字列が格納されます。
entPhysicalHardwareRev = E	物理エンティティのベンダー固有のハードウェア リビジョン ストリング。コンポーネント本体に実際に印刷されているハードウェア リビジョン ID（もしあれば）が、望ましい値です。
entPhysicalFirmwareRev =	物理エンティティのベンダー固有のファームウェア リビジョン ストリング。物理コンポーネントに特定のファームウェア プログラムが関連付けられていない場合、またはこの情報がエージェントに不明な場合は、このオブジェクトに長さゼロの文字列が格納されます。
entPhysicalSoftwareRev =<*user input*>	物理エンティティのベンダー固有のソフトウェア リビジョン ストリング。
entPhysicalSerialNum = SCA070400D8	物理エンティティのベンダー固有のシリアル番号ストリング。
entPhysicalMfgName = Cisco Systems Inc	この物理コンポーネントのメーカー名。

表 A-1　entPhysicalEntry の構成情報 （続き）

entPhysicalTEntries	entPhysicalTEntry の説明
entPhysicalModelName = 73-5916-02	この物理コンポーネントに関連付けられているベンダー固有のモデル ID ストリング。コンポーネント本体に印刷されているメーカーの部品番号が、望ましい値です。
entPhysicalAlias =	このオブジェクトは、ネットワーク マネージャが指定した物理エンティティのエイリアス名であり、物理エンティティの不揮発性ハンドルを提供します。 物理エンティティが初めてインスタンス化されるとき、そのエンティティに関連付けられている entPhysicalAlias の値は長さゼロの文字列に設定されます。ただし、エージェントは、この値を長さゼロの文字列ではなく、ローカルで一意のデフォルト値に設定します。
entPhysicalAssetID =	このオブジェクトは、ネットワーク マネージャが指定した、物理エンティティのユーザ割り当ての資産トラッキング ID であり、この情報を格納する不揮発性記憶装置を提供します。 物理エンティティが初めてインスタンス化されるとき、そのエンティティに関連付けられている entPhysicalAssetID の値は長さゼロの文字列に設定されます。
entPhysicalIsFRU = true(1)	このオブジェクトは、この物理エンティティが、ベンダーによって Field Replaceable Unit (FRU) と認識されているかどうかを示します。このオブジェクトには、値 true(1) が含まれます。よって、entPhysicalEntry は FRU です。 （注）　FRU　内に永続的に格納されるコンポーネントを表す entPhysicalEntries に対してはすべて、このオブジェクトに値 false(2) が戻されます。

次の出力例では、Cisco 7304 NSE-100 Daughter Card を搭載する Cisco 7304 シャーシに対する物理エンティティの包括ツリーを示しています。

entPhysicalTEntry.11

```
entPhysicalDescr.11 = Network Service Engine 100 Daughter Card
entPhysicalVendorType.11 = cevC7300Nse100Db
entPhysicalContainedIn.11 = 10
entPhysicalClass.11 = module(9)
entPhysicalParentRelPos.11 = 1
entPhysicalName.11 = Network Service Engine 100 Daughter Card 0
entPhysicalHardwareRev.11 = 5.0
entPhysicalFirmwareRev.11 =
entPhysicalSoftwareRev.11 =
entPhysicalSerialNum.11 = CAT07050DSN
entPhysicalMfgName.11 = Cisco Systems Inc
entPhysicalModelName.11 = 73-5673-07
entPhysicalAlias.11 =
entPhysicalAssetID.11 =
entPhysicalIsFRU.11 = false(2)
.
.
.
```

FRU ステータスの監視および設定

CISCO-ENTITY-FRU-CONTROL-MIB の cefcModuleTable 内のオブジェクトを調べて、電源装置やラインカードなどの Field Replaceable Unit (FRU) の管理ステータスおよび動作ステータスを判別します。

- cefcModuleAdminStatus : FRU の管理ステータス。cefcModuleAdminStatus を使用して、FRU をイネーブルまたはディセーブルにします。
- cefcModuleOperStatus：FRU の現在の動作状態。

（注）　現在、CISCO-ENTITY-FRU-CONTROL-MIB は Cisco 7304 ラインカードをサポートしています。MIB に関するその他の制約事項については、「CISCO-ENTITY-FRU-CONTROL-MIB」（P. 3-21）を参照してください。

次の例では、entPhysicalIndex が 9 である Cisco 7304 ラインカードの cefcModuleTable のエントリを示しています。

```
cefcModuleEntry.entPhysicalIndex

cefcModuleEntry.9
cefcModuleAdminStatus = enabled(1)
cefcModuleOperStatus = ok(2)
cefcModuleResetReason = manual reset(5)
cefcModuleStatusLastChangeTime = 7714
```

SNMP トラップの生成

ここでは、ルータ上のイベントおよび状態に応答して生成される SNMP トラップに関する情報を説明します。また、どのホストがトラップを受信するかを特定する方法を説明します。

- トラップを受信するホストの特定
- 設定の変更
- 環境状態
- FRU ステータスの変更

トラップを受信するホストの特定

CLI または SNMP を使用して、SNMP 通知を受信するホストを特定し、ホストが受信する通知タイプ（トラップまたはインフォーム）を指定できます。CLI の使用方法については、「通知のイネーブル化」（P. 4-2）を参照してください。SNMP を使用してこの情報を設定するには、次の MIB オブジェクトを使用します。

以下のものを含む SNMP-NOTIFICATION-MIB オブジェクトを使用して、ターゲット ホストを選択し、これらのホストに対して生成される通知タイプを指定します。

- snmpNotifyTable：ホストおよび通知タイプを選択するオブジェクトが含まれています。
  - snmpNotifyTag は、SNMP 通知を受信するホストを特定するために使用する任意のオクテット ストリング（タグ値）です。ターゲット ホストに関する情報は、snmpTargetAddrTable (SNMP-TARGET-MIB) で定義され、各ホストは、それぞれに対応する 1 つまたは複数のタグ値をもちます。snmpTargetAddrTable のホストがこの snmpNotifyTag 値と同じタグ値をもつ場合は、snmpNotifyType によって指定される通知タイプを受信するホストとして選択されます。

  - snmpNotifyType は、送信する SNMP 通知タイプで、trap(1) または inform(2) です。
- snmpNotifyFilterProfileTable および snmpNotifyFilterTable：これらのテーブルのオブジェクトを使用して、ターゲット ホストに送信する通知タイプを制限する通知フィルタを作成します。

SNMP-TARGET-MIB オブジェクトを使用して、通知を受信するホストに関する情報を設定します。

- snmpTargetAddrTable：SNMP 通知を受信するホストのアドレスを転送します。各エントリは、ホスト アドレスに関する情報（タグ値の一覧を含む）を提供します。
  - snmpTargetAddrTagList：ホスト　アドレスに対応する一連のタグ値。　ホストのタグ値が snmpNotifyTag と一致する場合には、snmpNotifyType で定義される通知タイプを受信するホストとして選択されます。
- snmpTargetParamsTable：SNMP 通知を生成するときに使用する SNMP パラメータです。

該当する MIB 内の通知イネーブル オブジェクトを使用して、特定の SNMP トラップをイネーブルおよびディセーブルにします。たとえば、mplsLdpSessionUp トラップまたは mplsLdpSessionDown トラップを生成するには、MPLS-LDP-MIB　オブジェクト　mplsLdpSessionUpDownTrapEnable　を enabled(1) に設定する必要があります。

設定の変更

エンティティ トラップがイネーブルの場合、ルータは次のいずれかのテーブル内の情報が変更されたとき（ルータの設定が変更されたとき）に、entConfigChange トラップ（ENTITY-MIB）を生成します。

- entPhysicalTable
- entAliasMappingTable
- entPhysicalContainsTable

（注）　設定の変更を追跡する管理アプリケーションは、entLastChangeTime (ENTITY-MIB) 値を必要に応じてチェックして、スロットリングまたは伝送ロスによって失われた entConfigChange トラップの有無を確認しなければなりません。

設定の変更に関するトラップのイネーブル化

ルータの設定を変更すると必ず entConfigChange トラップが生成されるようにルータを設定するには、CLI から次のコマンドを入力します。トラップをディセーブルにするには、コマンドの **no** 形式を使用します。

```
Router(config)# snmp-server enable traps entity
Router(config)# no snmp-server enable traps entity
```

環境状態

CISCO-ENVMON-MIB は、次の通知を送信して、ルータの環境センサーが検出する状態について警告を発します。

- ciscoEnvMonShutdownNotification：ルータをシャットダウンしようとするときに送信されます。
- ciscoEnvMonTemperatureNotification：温度が通常の範囲外になったときに送信されます。
- ciscoEnvMonRedundantSupplyNotification：冗長電源入力モジュールが故障したときに送信されます。

環境トラップのイネーブル化

環境状態に関するトラップが生成されるようにルータを設定するには、CLI から次のコマンドを入力します。トラップをディセーブルにするには、コマンドの **no** 形式を使用します。

```
Router(config)# snmp-server enable traps envmon
Router(config)# no snmp-server enable traps envmon
```

SNMP を使用して環境トラップをイネーブルにするには、該当する通知イネーブル オブジェクトを true(1) に設定します。たとえば、ciscoEnvMonEnableShutdownNotification は、シャットダウン通知をイネーブルにします。環境トラップをディセーブルにするには、通知オブジェクトを false(2) に設定します。

FRU ステータスの変更

FRU トラップがイネーブルの場合、ルータは FRU ステータスの変更に応答して、次のトラップを生成します。これらのトラップの詳細は、CISCO-ENTITY-FRU-CONTROL-MIB を参照してください。

- cefcModuleStatusChange：FRU に変更の動作ステータス（cefcModuleOperStatus）。
- cefcFRUInserted：FRU はシャーシに取り付けられています。このトラップは、FRU とその取り付け先コンテナの entPhysicalIndex を表します。
- cefcFRURemoved：FRU はシャーシから取り外されています。このトラップは、FRU と FRU が取り外されたコンテナの entPhysicalIndex を表します。

FRU トラップのイネーブル化

FRU イベントのトラップが生成されるようにルータを設定するには、CLI から次のコマンドを入力します。トラップをディセーブルにするには、コマンドの **no** 形式を使用します。

```
Router (config)# snmp-server enable traps fru-ctrl
Router(config)# no snmp-server enable traps fru-ctrl
```

SNMP を使用して FRU トラップをイネーブルにするには、cefcMIBEnableStatusNotification を true(1) に設定します。FRU トラップをディセーブルにするには、cefcMIBEnableStatusNotification を false(2) に設定します。

ルータ インターフェイスの監視

ここでは、ルータ インターフェイスのステータスを監視して、インターフェイス上のサービスに影響する可能性がある問題または状態があるかどうかを確認する方法を説明します。インターフェイスがダウンしているかどうか、あるいは、インターフェイスに問題が発生しているかどうかを判断するには、次のタスクを実行します。

インターフェイスの動作ステータスおよび管理ステータスのチェック

インターフェイスのステータスをチェックするには、インターフェイスに関する次の IF-MIB オブジェクトを確認します。

- ifAdminStatus：インターフェイスの適切な管理設定状態。ifAdminStatus を使用して、インターフェイスをイネーブルまたはディセーブルにします。
- ifOperStatus：インターフェイスの現在の動作ステータス。

linkDown トラップおよび linkUp トラップの監視

インターフェイスがダウンしているかどうかを判断するには、インターフェイスの linkDown トラップおよび linkUp トラップを監視します。これらのトラップをイネーブルにする方法については、「インターフェイスの linkUp トラップおよび linkDown トラップのイネーブル化」（P. A-11）を参照してください。

- linkDown：インターフェイスがダウンしているか、またはダウン寸前であることを示します。
- linkUp：インターフェイスがダウン状態ではないことを示します。

インターフェイスの linkUp トラップおよび linkDown トラップのイネーブル化

ルータ インターフェイスの状態が Up（作動可能）または Down（作動不能）に変わったときに通知を送信するように SNMP を設定するには、次の手順を実行して linkUp トラップおよび linkDown トラップをイネーブルにします。

ステップ 1　次の CLI コマンドを発行して、インターフェイスの linkUp トラップ および linkDown トラップのほとんどをイネーブルにします。これらのトラップをすべてイネーブルにする必要はありません。

```
Router(config)# snmp-server enable traps snmp linkdown linkup
```

ステップ 2　各インターフェイスの ifLinkUpDownTrapEnable object (IF-MIB ifXTable) の設定を確認して、linkUp トラップおよび linkDown トラップがそのインターフェイスに対してイネーブルか、ディセーブルかを判断します。

ステップ 3　インターフェイスの linkUp トラップおよび linkDown トラップをイネーブルにするには、ifLinkUpDownTrapEnable を enabled(1) に設定します。インターフェイスの最下位レイヤの linkDown トラップのみを送信するようにルータを設定する方法は、「linkDown トラップの SNMP トラップのフィルタリング機能」（P. A-12）を参照してください。

ステップ 4　linkUp トラップおよび linkDown トラップの Internet Engineering Task Force (IETF) 標準をイネーブルにするには、次のコマンドを発行します（IETF 標準は、RFC 2233 に基づきます）。

```
Router(config)# snmp-server trap link ietf
```

ステップ 5　ATM サブインターフェイス上の linkUp トラップおよび linkDown トラップをイネーブルにするには、次のコマンドを発行します。

```
Router(config)# snmp-server enable traps atm subif
```

ステップ 6　ATM 相手先固定接続（PVC）上の linkUp トラップおよび linkDown トラップをイネーブルにするには、次のコマンドを発行します。最初のコマンドで、**interval** は一連のトラップ間の最小間隔を指定し、**fail-interval** は失敗したタイムスタンプを保存する最小間隔を指定します。

```
Router(config)# snmp-server enable traps atm pvc interval seconds fail-interval seconds
Router(config)# interface atm slot/subslot/port
Router(config-if)# pvc vpi/vci
Router(config-if-atm-vc)# oam-pvc manage
```

ステップ 7　トラップをディセーブルにするには、該当するコマンドの **no** 形式を使用します。

linkDown トラップの SNMP トラップのフィルタリング機能

SNMP トラップのフィルタリング機能を使用して、linkDown トラップをフィルタします。これにより、メイン インターフェイスがダウンした場合に限り、SNMP は linkDown トラップを送信できます。インターフェイスがダウンすると、そのサブインターフェイスもすべてダウンします。その結果、各サブインターフェイスの linkDown トラップが大量に発生します。SNMP トラップのフィルタリング機能は、このようなサブインターフェイスのトラップをフィルタして除外します。

デフォルトでは、SNMP トラップのフィルタリング機能はオフになっています。この機能をイネーブルにするには、次の CLI コマンドを発行します。この機能をディセーブルにするには、コマンドの **no** 形式を使用します。

```
[no] snmp ifmib trap throttle
```

QoS の監視

ここでは、ルータ上で SNMP を使用して QoS の設定情報および統計情報にアクセスする例を示します。この項の構成は、次のとおりです。

目的および利点

従来は、QoS 設定情報および統計情報にアクセスするには、CLI から **show** コマンドを入力する方法しかありませんでした。

管理機能の拡張によって、SNMP を使用してルータ上の QoS 設定情報および統計情報にアクセスできるようになりました。つまり、QoS 情報を収集および保存して、管理アプリケーションで使用することができます。また、バルク ファイル転送を使用して別のシステムに情報をコピーすることもできます。

QoS に使用される MIB

- CISCO-CLASS-BASED-QOS-MIB

QoS の設定

QoS を設定するには、コマンドライン インターフェイス（CLI）を使用します。詳しい設定方法については、『Cisco 7300 Internet Series Router Software Configuration Guide』の「Configuring Quality of Service」を参照してください。

QoS 設定情報および統計情報へのアクセス

CISCO-CLASS-BASED-QOS-MIB は、QoS 設定情報および統計情報へのアクセスを提供します。SNMP を使用してルータに QoS を設定することはできませんが、CLI を使用して設定した QoS 設定情報に SNMP を使用してアクセスすることはできます。

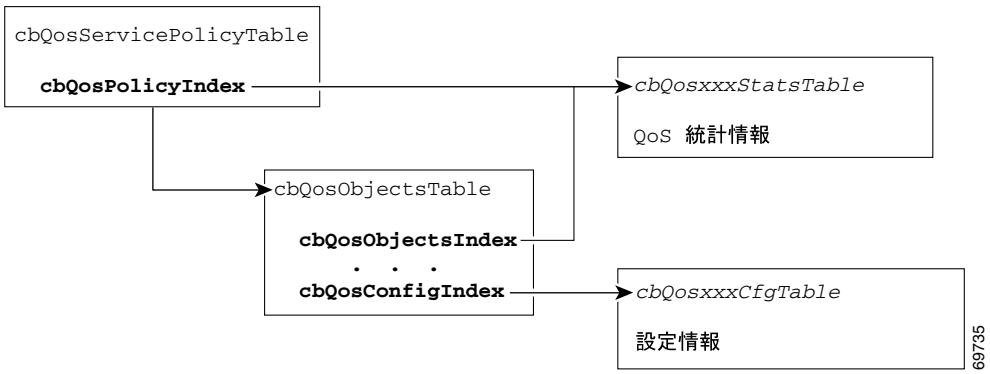
QoS インデックス

QoS 設定情報および統計情報にアクセスするときに使用するインデックスは、次のとおりです。

- cbQosPolicyIndex：インターフェイスに付加されたポリシー マップを識別する、システムによって割り当てられたインデックス（ポリシー マップは、インターフェイスに付加する時点では、サービス ポリシーといいます）。
- cbQosObjectsIndex：QoS 機能の固有の実行時インスタンス（たとえば、ポリシー マップ、クラス マップ、match 文、機能アクションなど）を識別する、システムによって割り当てられたインデックス。
- cbQosConfigIndex：QoS 機能の固有の設定（たとえば、クラス マップ、ポリシング アクションなど）を識別する、システムによって割り当てられたインデックス。同じ設定を使用する QoS オブジェクトは、同じ cbQosConfigIndex を共有するので注意してください。
- cbQosREDValue：WRED（重み付けランダム早期検出）アクションの IP 優先順位または IP Differentiated Services Code Point (DSCP)。各 RED クラスの設定情報および統計情報に対応するインデックスとして使用します。

図 A-1 では、これらのインデックスを使用して QoS 設定情報および統計情報にアクセスする仕組みを示しています。

図 A-1　Cisco 7304 ルータの QoS インデックス



特定の QoS 機能について QoS 設定情報および統計情報にアクセスする手順は、次のとおりです。

1. cbQosServicePolicyTable を検索し、その機能が使用されているポリシーに割り当てられた cbQosPolicyIndex を調べます。
2. cbQosPolicyIndex を使用して cbQosObjectsTable にアクセスし、QoS 機能に割り当てられた cbQosObjectsIndex および cbQosConfigIndex を調べます。
   - cbQosConfigIndex を使用して、設定テーブル（cbQosxxxCfgTable）で機能に関する情報にアクセスします。
   - cbQosPolicyIndex および cbQosObjectsIndex を使用して、QoS 統計テーブル（cbQosxxxStatsTable）で QoS 機能に関する情報にアクセスします。

QoS 設定の設定値の例

ここでは、QoS 設定の設定値を CISCO-CLASS-BASED-QOS-MIB テーブルに保存する方法と、QoS オブジェクト別にグループ化した情報を示します。ただし、SNMP クエリーによって出力される実際の QoS 情報は、次のような形式になります。この出力はすべての QoS 情報の一部に過ぎません。

7304# getmany -v3 10.86.0.94 test-user ciscoCBQosMIB

```
cbQosIfType.9583 = mainInterface(1)
cbQosIfType.9619 = mainInterface(1)
cbQosPolicyDirection.9583 = input(1)
cbQosPolicyDirection.9619 = output(2)
cbQosIfIndex.9583 = 3
cbQosIfIndex.9619 = 3
cbQosFrDLCI.9583 = 0
cbQosFrDLCI.9619 = 0
cbQosAtmVPI.9583 = 0
cbQosAtmVPI.9619 = 0
cbQosAtmVCI.9583 = 0
cbQosAtmVCI.9619 = 0
cbQosConfigIndex.9583.9583 = 9457
cbQosConfigIndex.9583.9585 = 9451
cbQosConfigIndex.9583.9587 = 9455
cbQosConfigIndex.9583.9589 = 9459
cbQosConfigIndex.9583.9591 = 9445
cbQosConfigIndex.9583.9593 = 9449
cbQosConfigIndex.9583.9595 = 9461
cbQosConfigIndex.9583.9597 = 1025
cbQosConfigIndex.9583.9599 = 1027
cbQosConfigIndex.9619.9619 = 9609
cbQosConfigIndex.9619.9621 = 9601
cbQosConfigIndex.9619.9623 = 9605
cbQosConfigIndex.9619.9625 = 9607
cbQosConfigIndex.9619.9627 = 9611
cbQosConfigIndex.9619.9629 = 1025
cbQosConfigIndex.9619.9631 = 1027
cbQosObjectsType.9583.9583 = policymap(1)
cbQosObjectsType.9583.9585 = classmap(2)
cbQosObjectsType.9583.9587 = matchStatement(3)
cbQosObjectsType.9583.9589 = police(7)
cbQosObjectsType.9583.9591 = classmap(2)
cbQosObjectsType.9583.9593 = matchStatement(3)
cbQosObjectsType.9583.9595 = police(7)
cbQosObjectsType.9583.9597 = classmap(2)
cbQosObjectsType.9583.9599 = matchStatement(3)
cbQosObjectsType.9619.9619 = policymap(1)
cbQosObjectsType.9619.9621 = classmap(2)
cbQosObjectsType.9619.9623 = matchStatement(3)
cbQosObjectsType.9619.9625 = matchStatement(3)
cbQosObjectsType.9619.9627 = queueing(4)
cbQosObjectsType.9619.9629 = classmap(2)
cbQosObjectsType.9619.9631 = matchStatement(3)
cbQosParentObjectsIndex.9583.9583 = 0
cbQosParentObjectsIndex.9583.9585 = 9583
cbQosParentObjectsIndex.9583.9587 = 9585
cbQosParentObjectsIndex.9583.9589 = 9585
cbQosParentObjectsIndex.9583.9591 = 9583
cbQosParentObjectsIndex.9583.9593 = 9591
cbQosParentObjectsIndex.9583.9595 = 9591
cbQosParentObjectsIndex.9583.9597 = 9583
cbQosParentObjectsIndex.9583.9599 = 9597
cbQosParentObjectsIndex.9619.9619 = 0
cbQosParentObjectsIndex.9619.9621 = 9619
cbQosParentObjectsIndex.9619.9623 = 9621
cbQosParentObjectsIndex.9619.9625 = 9621
cbQosParentObjectsIndex.9619.9627 = 9621
cbQosParentObjectsIndex.9619.9629 = 9619
cbQosParentObjectsIndex.9619.9631 = 9629
cbQosPolicyMapName.9457 = car
cbQosPolicyMapDesc.9457 =
cbQosCMName.1025 = class-default
cbQosCMDesc.1025 =
cbQosCMInfo.1025 = matchAny(3)
     . . .
```

次の出力は、QoS CLI の **show** コマンドの例です。

7304# show class-map

```
class-map match-any prec1_dscp16
match ip prec 1
match ip dscp 16
```

7304# show policy-map

```
policy-map cbwfq-wred
class prec1_dscp16
       bandwidth percent 20
       random-detect
       random-detect exponential-weighting-constant 10
       random-detect precedence 0 10 126 10
       random-detect precedence 1 11,126 10
       random-detect precedence 2 12,126 10
       random-detect precedence 3 13,126 10
       random-detect precedence 4 14,126 10
       random-detect precedence 5 15,126 10
       random-detect precedence 6 16,126 10
       random-detect precedence 7 17,126 10
 interface gi0/1
service-policy output cbwfq-wred
policy-map car
       class prec1
          police 20000000 24000 24000 conform-action  set-dscp-transmit 5
exceed-action drop
       class dscp16
          police 20000000 24000 24000 conform-action transmit  exceed-action
transmit
 interface pos4/0
service-policy input car
```

この QoS 設定例について、次の点に注意してください。

- インターフェイスに付加されていないポリシー　マップは、SNMP　データに含まれず、**show policy-map interface** コマンドで表示されません。したがって、pm-1Meg は表示されますが、pm1 は表示されません
- SNMP データには、常にデフォルトのクラス マップが含まれています。
- アクションが定義されていないクラス マップは、SNMP データには含まれません。

次の出力では、Cisco 7304 インターフェイスに適用される MIB テーブルに保存された RED 設定情報の例を示しています。

```
cbQosREDCfgExponWeight.9776 = 10
cbQosREDCfgMeanQsize.9776 = 0
cbQosREDCfgDscpPrec.9776 = precedence(1)
cbQosREDCfgMinThreshold.9776.0 = 10
cbQosREDCfgMinThreshold.9776.1 = 11
cbQosREDCfgMinThreshold.9776.2 = 12
cbQosREDCfgMinThreshold.9776.3 = 13
cbQosREDCfgMinThreshold.9776.4 = 14
cbQosREDCfgMinThreshold.9776.5 = 15
cbQosREDCfgMinThreshold.9776.6 = 16
cbQosREDCfgMinThreshold.9776.7 = 17
cbQosREDCfgMaxThreshold.9776.0 = 126
cbQosREDCfgMaxThreshold.9776.1 = 126
cbQosREDCfgMaxThreshold.9776.2 = 126
cbQosREDCfgMaxThreshold.9776.3 = 126
cbQosREDCfgMaxThreshold.9776.4 = 126
cbQosREDCfgMaxThreshold.9776.5 = 126
cbQosREDCfgMaxThreshold.9776.6 = 126
cbQosREDCfgMaxThreshold.9776.7 = 126
cbQosREDCfgPktDropProb.9776.0 = 10
cbQosREDCfgPktDropProb.9776.1 = 10
cbQosREDCfgPktDropProb.9776.2 = 10
cbQosREDCfgPktDropProb.9776.3 = 10
cbQosREDCfgPktDropProb.9776.4 = 10
cbQosREDCfgPktDropProb.9776.5 = 10
cbQosREDCfgPktDropProb.9776.6 = 10
cbQosREDCfgPktDropProb.9776.7 = 10
. . .
```

QoS の監視

ここでは、表 A-2 に示されている MIB テーブルの QoS 統計情報をチェックすることによって、ルータ上で QoS を監視する方法を説明します。カスタマーに課金するトラフィック量を決定する方法については、「カスタマーに課金するトラフィック」（P. A-22）を参照してください。

（注）　CISCO-CLASS-BASED-QOS-MIB には、CLI の **show** コマンド出力よりも詳しい情報が含まれる場合があります。

表 A-2　QoS 統計情報テーブル

QoS テーブル	統計情報
cbQosCMStatsTable	クラス マップ：QoS ポリシーの実行前および実行後のパケット数、バイト数、およびビット レート。廃棄されたパケット数およびバイト数。
cbQosMatchStmtStatsTable	match 文：QoS ポリシーを実行する前のパケット数、バイト数、およびビット レート。
cbQosPoliceStatsTable	ポリシング アクション：ポリシング アクションに適合、超過、および違反したパケット数、バイト数、およびビット レート。
cbQosQueueingStatsTable	キューイング：廃棄されたパケット数およびバイト数、およびキュー深度。
cbQosTSStatsTable	トラフィック シェーピング：遅延および廃棄されたパケット数およびバイト数、機能の状態、およびキュー サイズ。
cbQosREDClassStatsTable	ランダム早期検出：キューが満杯のとき廃棄されたパケット数およびバイト数、および送信されたパケット数およびバイト数。

QoS 統計の処理に関する考慮事項

ルータでは、ほとんどの QoS 統計情報について 64 ビット カウンタが維持されます。ただし、一部の QoS カウンタは、1 ビットのオーバーフロー フラグ付きの 32 ビット カウンタとして実装されています。次の図では、これらのカウンタは 33 ビット カウンタとして示されています。

カウンタの QoS 統計情報にアクセスする際、次の点に注意してください。

- SNMPv2c または SNMPv3 アプリケーション：*cbQosxxx64* MIB オブジェクトを使用して、QoS カウンタの 64 ビット全体にアクセスします。
- SNMPv1 アプリケーション：次のようにして MIB 内の QoS 統計情報にアクセスします。
  - *cbQosxxx* MIB オブジェクトを使用して、カウンタの下位 32 ビットにアクセスします。
  - *cbQosxxxOverflow* MIB オブジェクトを使用して、カウンタの上位 32 ビットにアクセスします。

QoS 統計情報の例

ここでは、**show** コマンドで QoS 統計情報を表示し、その情報を CISCO-CLASS-BASED-QOS-MIB テーブルに保存する方法を示します。

次の例では、見やすさを配慮して、一部のカウンタについては、実際には 3 つのオブジェクトとして実装されていても 1 つのオブジェクトで表しています。たとえば、cbQosCMPrePolicyByte は次のように実装されています。

```
cbQosCMPrePolicyByteOverflow
cbQosCMPrePolicyByte
cbQosCMPrePolicyByte64
```

（注）　実装上の機能として、一部の QoS 統計情報カウンタは、対応できる最大値に達する前にラップアラウンドする場合があります。

次の出力は、QoS クラス マップの統計情報の例です。

```
class-map dscp16
       match ip dscp 16
       class-map prec1
       match ip prec 1
       policy-map car
       class prec1
          police 20000000 24000 24000 conform-action  set-dscp-transmit 5 exce
ed-action drop
       class dscp16
police 20000000 24000 24000 conform-action transmit  exceed-action transmit
 interface atm3/0
pvc 1/99
service-policy input car

class-map prec2
       match ip prec 2
       class-map prec1
       match ip prec 1
       policy-map cbwfq
       class prec1
         bandwidth perc 40
         queue-limit 33
       class prec2
         bandwidth perc 50
         queue-limit 55
 interface atm3/0
pvc 1/99
service-policy output cbwfq
```

cbQosPoliceStatsTable

```
cbQosPoliceStatsEntry
    cbQosPoliceConformedPktOverflow.9535.9541 = 0
    cbQosPoliceConformedPkt.9535.9541 = 226961
    cbQosPoliceConformedPkt64.9535.9541 = 0x000037691
    cbQosPoliceConformedByte.9535.9541 = 230592376
    cbQosPoliceConformedByteOverflow.9535.9541 = 0
    cbQosPoliceConformedByte64.9535.9541 = 0x00dbe8f78
    cbQosPoliceConformedBitRate.9535.9541 = 0
    cbQosPoliceExceededPktOverflow.9535.9541 = 0
    cbQosPoliceExceededPkt64.9535.9541 = 0x000000000
    cbQosPoliceExceededByte.9535.9541 = 0
    cbQosPoliceExceededBitRate.9535.9541 = 0
    cbQosPoliceViolatedPktOverflow.9535.9541 = 0
    cbQosPoliceViolatedPkt.9535.9541 = 0
    cbQosPoliceViolatedByteOverflow.9535.9541 = 0
    cbQosPoliceViolatedByte.9535.9541 = 0
    cbQosPoliceViolatedBitRate.9535.9541 = 0
```

cbQosQueueingStatsTable

```
cbQosQueueingStatsEntry
    cbQosQueueingCurrentQDepth.9619.9627 = 0
    cbQosQueueingMaxQDepth.9619.9627 = 0
    cbQosQueueingDiscardByte.9619.9627 = 700700636
    cbQosQueueingDiscardPkt.9619.9627 = 697909
```

QoS アプリケーションの例

ここでは、CISCO-CLASS-BASED-QOS-MIB から情報を取り出して QoS 課金処理に使用するためのコード例を示します。課金アプリケーションを開発するときは、これらの例を参考にしてください。このコード例では、次の処理方法を示します。

- サービス ポリシーに関するカスタマー インターフェイスのチェック
- QoS 課金情報の取得

サービス ポリシーに関するカスタマー インターフェイスのチェック

ここでは、サービス ポリシーのあるカスタマー インターフェイスについて CISCO-CLASS-BASED-QOS-MIB をチェックし、QoS サービスの課金などのアプリケーション処理を行えるように、これらのインターフェイスをマークするアルゴリズムの例を示します。

このアルゴリズムでは、カスタマー インターフェイスごとに 2 つの SNMP **get-next** 要求を使用します。たとえば、ルータ上に 2000 個のカスタマー インターフェイスがある場合、それらのインターフェイスに対応付けられた送信および受信サービス ポリシーがあるかどうかを判断するには、4000 個の SNMP **get-next** 要求が必要になります。

（注）　このアルゴリズムは一例に過ぎません。実際に使用するアプリケーションでは、この例とは異なる処理が必要になる場合があります。

特定のカスタマーに対応するインターフェイスを調べるには、MIB をチェックします。カスタマー インターフェイスの送信方向および受信方向にサービス ポリシーが対応付けられているかどうかを示す 1 対のフラグを作成します。非カスタマー インターフェイスは TRUE でマークします（これらのインターフェイスについては、以降の処理は不要です）。

```
FOR each ifEntry DO
  IF (ifEntry represents a customer interface) THEN
     servicePolicyAssociated[ifIndex].transmit = FALSE;
     servicePolicyAssociated[ifIndex].receive = FALSE;
  ELSE
     servicePolicyAssociated[ifIndex].transmit = TRUE;
     servicePolicyAssociated[ifIndex].receive = TRUE;
  END-IF
END-FOR
```

cbQosServicePolicyTable を調べ、サービス ポリシーが付加されている各カスタマー インターフェイスをマークします。さらに、インターフェイスの方向もチェックします。

```
x = 0;
done = FALSE;
WHILE (!done)
  status = snmp-getnext (
           ifIndex = cbQosIfIndex.x,
           direction = cbQosPolicyDirection.x
  );
  IF (status != 'noError') THEN
     done = TRUE
  ELSE
     x = extract cbQosPolicyIndex from response;
     IF (direction == 'output') THEN
       servicePolicyAssociated[ifIndex].transmit = TRUE;
     ELSE
       servicePolicyAssociated[ifIndex].receive = TRUE;
     END-IF
  END-IF
END-WHILE
```

サービス ポリシーが付加されていないカスタマー インターフェイスの問題を処理します。

```
FOR each ifEntry DO
  IF (!servicePolicyAssociated[ifIndex].transmit) THEN
     Perform processing for customer interface without a transmit service policy.
  END-IF
  IF (!servicePolicyAssociated[ifIndex].receive) THEN
     Perform processing for customer interface without a receive service policy.
  END-IF
END-FOR
```

QoS 課金情報の取得

ここでは、CISCO-CLASS-BASED-QOS-MIB を使用して QoS 課金処理を行うアルゴリズムの例を示します。このアルゴリズムは、ポリシング後の入力および出力統計情報を定期的に取り出し、それらを組み合わせた結果を課金データベースに送信します。

このアルゴリズムでは、次の要求を使用しています。

- カスタマー インターフェイスごとに 1 つの SNMP **get** 要求：ifAlias を取得します。
- カスタマー インターフェイスごとに 2 つの SNMP **get-next** 要求：サービス ポリシー インデックスを取得します。
- ポリシーに含まれる各オブジェクトについて、カスタマー インターフェイスごとに 2 つの SNMP **get-next** 要求 : ポリシング後のバイト数を取得します。たとえば、ポリシーに 100 個のインターフェイスと 10 個のオブジェクトがある場合、このアルゴリズムでは 2000 個（2×100×10）の **get-next** 要求が必要になります。

（注）　このアルゴリズムは、情報として示しているに過ぎません。実際に使用するアプリケーションでは、この例とは異なる処理が必要になる場合があります。

カスタマー課金情報を設定します。

```
FOR each ifEntry DO
  IF (ifEntry represents a customer interface) THEN
     status = snmp-getnext (id = ifAlias.ifIndex);
     IF (status != 'noError') THEN
         Perform error processing.
     ELSE
        billing[ifIndex].isCustomerInterface = TRUE;
        billing[ifIndex].customerID = id;
        billing[ifIndex].transmit   = 0;
        billing[ifIndex].receive    = 0;
     END-IF
  ELSE
     billing[ifIndex].isCustomerInterface = FALSE;
  END-IF
END-FOR
```

課金情報を取得します。

```
x = 0;
done = FALSE;
WHILE (!done)
  response = snmp-getnext (
             ifIndex = cbQosIfIndex.x,
             direction = cbQosPolicyDirection.x
  );
  IF (response.status != 'noError') THEN
     done = TRUE
  ELSE
     x = extract cbQosPolicyIndex from response;
     IF (direction == 'output') THEN
        billing[ifIndex].transmit = GetPostPolicyBytes (x);
     ELSE
        billing[ifIndex].receive = GetPostPolicyBytes (x);
     END-IF
  END-IF
END-WHILE
```

課金のためポリシング後のバイト数を判断します。

```
GetPostPolicyBytes (policy)
  x = policy;
  y = 0;
  total = 0;
  WHILE (x == policy)
     response = snmp-getnext (type = cbQosObjectsType.x.y);
     IF (response.status == 'noError')
        x = extract cbQosPolicyIndex from response;
        y = extract cbQosObjectsIndex from response;
        IF (x == policy AND type == 'classmap')
           status = snmp-get (bytes = cbQosCMPostPolicyByte64.x.y);
           IF (status == 'noError')
                   total += bytes;
           END-IF
        END-IF
     END-IF
  END-WHILE
RETURN total;
```

カスタマーに課金するトラフィック

ここでは、SNMP QoS 情報を使ってカスタマーに課金するトラフィック合計を算出する方法を説明します。また、インターフェイスに付加された QoS サービス ポリシーがそのインターフェイスのポリシング トラフィックであることを示すシナリオについても説明します。

この項の構成は、次のとおりです。

入出力インターフェイスのカウント

ルータは、入出力インターフェイスで送受信されたパケット数やバイト数の情報を保持します。QoS サービス ポリシーがインターフェイスに付加されると、ルータはインターフェイス上のトラフィックにポリシーのルールを適用し、インターフェイスのパケットとバイトのカウントを増やします

次の CISCO-CLASS-BASED-QOS-MIB オブジェクトにより、インターフェイスのカウントがわかります。

- cbQosCMDropPkt および cbQosCMDropByte (cbQosCMStatsTable)：サービス ポリシーで設定された制限を超えたために廃棄されたパケットとバイトの合計数。このカウントに含まれるのは、サービス ポリシーの制限を超えたために廃棄されたパケットとバイトだけです。他の理由で廃棄されたパケットとバイトはカウントに含まれません。
- cbQosPoliceConformedPkt および cbQosPoliceConformedByte (cbQosPoliceStatsTable)：サービス ポリシーの制限内であったために伝送されたパケットとバイトの合計数。

カスタマーに課金するトラフィック合計の算出方法

あるインターフェイスで特定のカスタマーに課金できるトラフィックを算出する手順は、次のとおりです。

ステップ 1　そのインターフェイスでカスタマーに適用するサービス ポリシーを調べます。

ステップ 2　カスタマーのトラフィックを定義するサービス ポリシーとクラス マップのインデックス値を調べます。この情報は、次のステップで必要になります。

ステップ 3　カスタマーの cbQosPoliceConformedPkt オブジェクト（cbQosPoliceStatsTable）にアクセスし、インターフェイスでこのカスタマーに課金できるトラフィックを算出します。

ステップ 4　（オプション）カスタマーの cbQosCMDropPkt オブジェクト（cbQosCMStatsTable）にアクセスし、このカスタマーのトラフィックのうちサービス ポリシーの制限を超えたために廃棄されたトラフィックを算出します。

QoS トラフィック ポリシングを示すシナリオ

ここでは、あるインターフェイスで特定のカスタマーに課金できるトラフィックを算出するための SNMP QoS 統計情報の使用を示すシナリオについて説明します。また、インターフェイスのトラフィックにサービス ポリシーが適用された場合、パケット カウントがどのように変わるかについても示します。

シナリオを作成する手順は、次のとおりです。それぞれの手順については後で説明します。

1. サービス ポリシーを作成し、インターフェイスに付加します。
2. サービス ポリシーがインターフェイスのトラフィックに適用される前のパケット カウントを確認します。
3. **ping** コマンドを発行し、そのインターフェイスでトラフィックを生成します。このトラフィックにはサービス ポリシーが適用されるので注意してください。
4. サービス ポリシーが適用された後のパケット カウントを確認し、カスタマーに課金するトラフィックを算出します。
   - 準拠しているパケット：サービス ポリシーで設定した範囲内のパケット数で、カスタマーに課金できます。
   - 超過したパケットまたは廃棄されたパケット：サービス ポリシーの範囲外であったために伝送されなかったパケットの数。これらのパケットはカスタマーに課金されません。

（注）　上記のシナリオでは、Cisco 7304 ルータは中間デバイスとして使用されています。つまり、トラフィックは他の場所で発生し、他のデバイスに到達します。

サービス ポリシーの設定

このシナリオでは、次のポリシー マップ設定を使用します。ポリシー マップの作成方法については、『Cisco 7300 Internet Series Router Software Configuration Guide』の「Configuring Quality of Service」を参照してください。

```
policy-map police-out
  class BGPclass
    police 8000 1000 2000 conform-action transmit exceed-action drop

interface GigabitEthernet1/0/0.10
 description VLAN voor klant
 encapsulation dot1Q 10
 ip address 10.0.0.17 255.255.255.248
 service-policy output police-out
```

サービス ポリシーを適用する前のパケット カウント

次の CLI 出力および SNMP 出力は、サービス ポリシーが適用される前のインターフェイスの出力トラフィックを示します。

CLI コマンド出力の例

```
7304# show policy-map interface g6/0/0.10

GigabitEthernet6/0/0.10

Service-policy output: police-out

   Class-map: BGPclass (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: access-group 101
      Police:
        8000 bps, 1000 limit, 2000 extended limit
        conformed 0 packets, 0 bytes; action: transmit
        exceeded 0 packets, 0 bytes; action: drop

   Class-map: class-default (match-any)
      4 packets, 292 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
      Output queue: 0/8192; 2/128 packets/bytes output, 0 drops
```

SNMP 出力の例

```
7304# getone -v2c 10.86.0.63 public ifDescr.65
ifDescr.65 = GigabitEthernet6/0/0.10-802.1Q vLAN subif
```

トラフィックの生成

次の一連の **ping** コマンドを使うと、トラフィックが生成されます。

```
7304# ping
Protocol [ip]:
Target IP address: 10.0.0.18
Repeat count [5]: 99
Datagram size [100]: 1400
Timeout in seconds [2]: 1
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.

Sending 100, 1400-byte ICMP Echos to 10.0.0.18, timeout is 1 seconds:
.!!.!..!.!..!.!.!..!.!..!.!..!.!.!..!.!..!.!..!.!.!..!.!..!.!..!.!.!..
!.!..!.!..!.!.!..!.!..!.!..!.!
Success rate is 42 percent (42/100), round-trip min/avg/max = 1/1/1 ms
```

サービス ポリシーを適用した後のパケット カウント

ping コマンドを使用してトラフィックを生成した後は、**police** コマンドで設定した専用アクセス レート（CAR）を超過したパケット数と CAR に準拠したパケット数を確認します。

- 42 パケットがポリシング レートに準拠し、伝送されました。
- 57 パケットがポリシング レートを超過し、廃棄されました。

次の CLI 出力および SNMP 出力は、サービス ポリシーが適用された後のインターフェイスのカウントを示します（準拠したパケットと超過したパケットの数は太字で示されています）。

CLI コマンド出力

```
7304# show policy-map interface g6/0/0.10

FastEthernet6/0/0.10

Service-policy output: police-out

   Class-map: BGPclass (match-all)
      198 packets, 281556 bytes
      30 second offered rate 31000 bps, drop rate 11000 bps
      Match: access-group 101
      Police:
        8000 bps, 1000 limit, 2000 extended limit
        conformed 42 packets, 59892 bytes; action: transmit
        exceeded 57 packets, 81282 bytes; action: drop

   Class-map: class-default (match-any)
      15 packets, 1086 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
      Output queue: 0/8192; 48/59940 packets/bytes output, 0 drops
```

SNMP 出力

```
7304# getmany -v2c 10.86.0.63 public ciscoCBQosMIB
        . . .
      cbQosCMDropPkt.1143.1145 = 57
        . . .
      cbQosPoliceConformedPkt.1143.1151 = 42
        . . .
```