MIB の使用方法

この章では、Cisco 10000 シリーズ ESR 上で SNMP（簡易ネットワーク管理プロトコル）を使用して次の機能を実行する方法について説明します。ルータで SNMP を使用して、ルーティング テーブルのエントリをポーリングする際に生じる問題の回避方法については、「CPU 使用率の負荷の抑制」（p.2-7）を参照してください。

ESR 物理エンティティの管理

ここでは、SNMP を使用して Cisco 10000 シリーズ ESR 上の物理エンティティ（コンポーネント）を管理する方法について説明します。具体的な方法は次のとおりです。

ラインカードをシャーシに挿入する前に SNMP を使用してラインカードの動作特性を事前設定する手順については、「ラインカードの事前プロビジョニング」（p.A-18）を参照してください。

目的および利点

Cisco 10000 の SNMP が実装した物理エンティティ管理機能は以下のとおりです。

- シャーシ内の物理エンティティを、各エンティティと他のすべてのエンティティとの関係について記述する包含ツリーに編成します。
- 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。
- 資産タギングに関する資産情報を提供します。
- 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情報を提供します。

物理エンティティの管理に使用する MIB

- CISCO-ENTITY-ASSET-MIB — ENTITY-MIB の entPhysicalTable に示された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 Management Information Base（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、ソフトウェア、ファームウェア情報など）を提供します。
- CISCO-ENTITY-FRU-CONTROL-MIB — ENTITY-MIB の entPhysicalTable に示された FRU（電源装置やラインカードなど）の管理ステータスと動作ステータスを監視および設定するために使用するオブジェクトが含まれています。

（注）　現在のところ、CISCO-ENTITY-FRU-CONTROL-MIB はラインカードのみをサポートしています。

- CISCO-ENTITY-VENDORTYPE-OID-MIB — Cisco 10000 シリーズ ESR のすべての物理エンティティに関する Object Identifier（OID; オブジェクト ID）が含まれています。
- CISCO-ENVMON-MIB — 環境センサ（電圧、温度、ファン、および電源装置）のステータス情報が含まれています。たとえば、この MIB はシャーシの内部および吸気口の温度を報告します。
- ENTITY-MIB — Cisco 10000 シリーズ ESR 上の物理エンティティをの管理に関する情報が含まれています。また、これらのエンティティを階層構造および相互関係を記述する包含ツリーに編成します。この MIB の内容は、次のとおりです。
  - entPhysicalTable は、Cisco 10000 シリーズ ESR の各物理コンポーネント（エンティティ）について記述しています。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、シャーシのエンティティの階層構造に適合させる方法を記述しています。

    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。

- entAliasMappingTable は、各物理ポートの entPhysicalIndex 値をそれに対応する IF-MIB の ifTable 上の ifIndex 値にマッピングします。
- entPhysicalContainsTable は、シャーシ内の物理エンティティ間の関係を示しています。このテーブルは、物理エンティティごとに各エンティティの子オブジェクトに関する entPhysicalIndex を示しています。

インベントリ管理の実行

ENTITY-MIB の entPhysicalTable 上の MIB Walk を実行して、Cisco 10000 シリーズ ESR のエンティティに関する情報を入手します。

図 A-1 から図 A-4 では、entPhysicalTable のエントリがエンティティに関する情報を提供する方法について示しています。

entPhysicalTable のエントリに関する注記

ENTITY-MIB の entPhysicalTable 上のエントリを調べる際は、以下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 テーブル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 値を持つ同タイプのエンティティ（シャーシ スロットおよびラインカード ポートなど）間の相対的な位置を示します（図 A-4 を参照）。

entPhysicalTable のエントリの例

ここでは、情報が entPhysicalTable に保存される手順について図で示します。entPhysicalTable のエントリを調べて、Cisco 10000 シリーズ ESR のコンフィギュレーションを確認することができます。

図 A-1 に、Cisco 10000 シリーズ ESR シャーシのスロット 1 に挿入されたギガビット イーサネット ラインカードおよびこのラインカードのポートに関する ENTITY-MIB の entPhysicalTable のエントリを示します。

図 A-1　　シャーシ エンティティに関する entPhysicalTable のエントリ

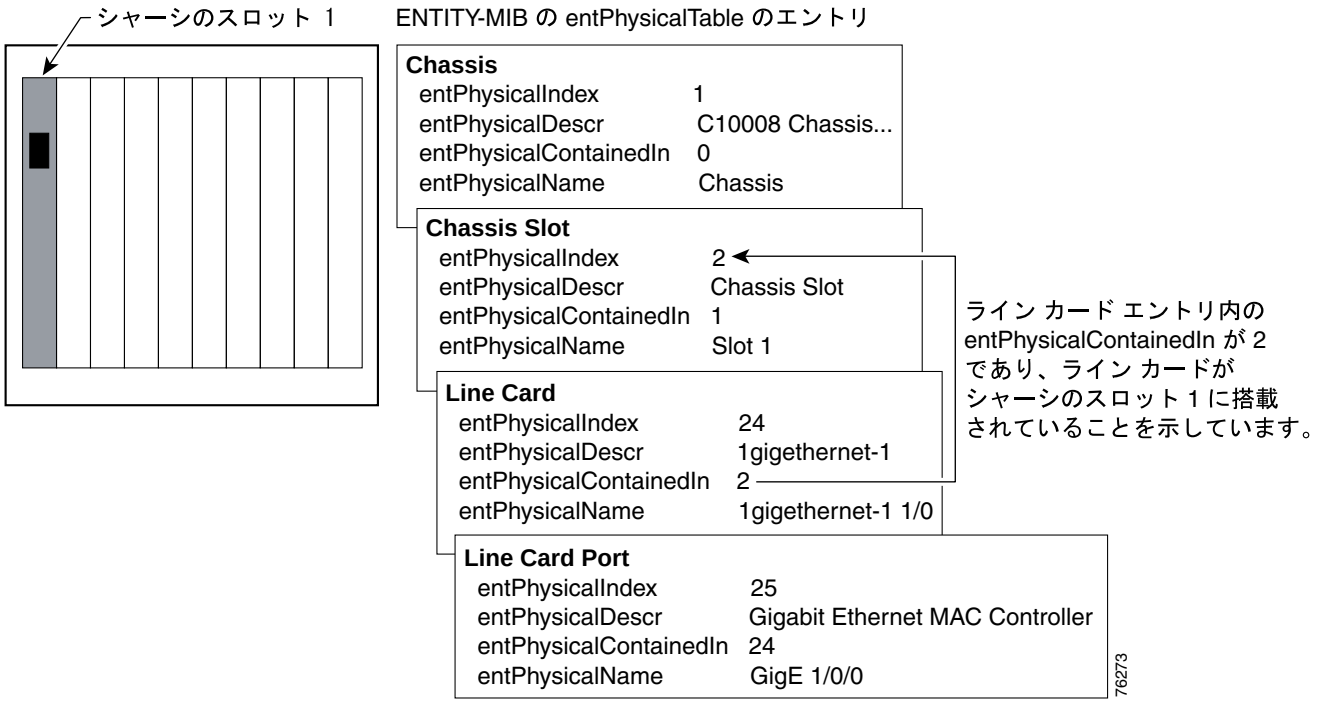


図 A-2 に、図 A-3 および図 A-4 で示したエンティティに関する entPhysicalTable のエントリの例を示します。

図 A-2　entPhysicalTable のエントリの例

entPhysicalTable
entPhysicalEntry.*entPhysicalIndex*

entPhysicalContainedIn は、エンティティの親の **entPhysicalIndex** です。

```
entPhysicalEntry.1
  entPhysicalDescr            C10008 chassis, Hw Serial#:...
  entPhysicalVendorType       CiscoModules.3.1.3.303
  entPhysicalContainedIn      0
  entPhysicalClass            chassis(3)
  entPhysicalParentRelPos     -1

entPhysicalEntry.2
  entPhysicalDescr            Chassis Slot
  entPhysicalVendorType       CiscoModules.3.1.5.86
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     1
  entPhysicalName             slot 1

entPhysicalEntry.3
  entPhysicalDescr            Chassis Slot
  entPhysicalVendorType       CiscoModules.3.1.5.86
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     2
  entPhysicalName             slot 2

entPhysicalEntry.4
  entPhysicalDescr            Chassis Slot
  entPhysicalVendorType       CiscoModules.3.1.5.86
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     3
  entPhysicalName             slot 3

entPhysicalEntry.5
  entPhysicalDescr            Chassis Slot
  entPhysicalVendorType       CiscoModules.3.1.5.86
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     4
  entPhysicalName             slot 4

entPhysicalEntry.6
  entPhysicalDescr            Chassis Slot
  entPhysicalVendorType       CiscoModules.3.1.5.86
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     5
  entPhysicalName             slot A
      . . .

entPhysicalEntry.12
  entPhysicalDescr            Power Supply Container
  entPhysicalVendorType       CiscoModules.3.1.5.87
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     12

entPhysicalEntry.13
  entPhysicalDescr            Power Supply
  entPhysicalVendorType       CiscoModules.3.1.6.56
  entPhysicalContainedIn      12
  entPhysicalClass            powerSupply(6)
  entPhysicalParentRelPos     1
      . . .
entPhysicalEntry.15
  entPhysicalDescr            Fan Tray Container
  entPhysicalVendorType       CiscoModules.3.1.5.88
  entPhysicalContainedIn      1
  entPhysicalClass            container(5)
  entPhysicalParentRelPos     12

entPhysicalEntry.16
  entPhysicalDescr            Fan Tray
  entPhysicalVendorType       CiscoModules.3.1.7.25
  entPhysicalContainedIn      15
  entPhysicalClass            module(9)
  entPhysicalParentRelPos     1
      . . .
entPhysicalEntry.20
  entPhysicalDescr            Route Processor
  entPhysicalVendorType       CiscoModules.3.1.9.5.29
  entPhysicalContainedIn      6
  entPhysicalClass            module(9)
  entPhysicalParentRelPos     1

entPhysicalEntry.21
  entPhysicalDescr            Forwarding Processor
  entPhysicalVendorType       CiscoModules.3.1.9.5.30
  entPhysicalContainedIn      20
  entPhysicalClass            module(9)
  entPhysicalParentRelPos     1
```

76274

図 A-2　entPhysicalTable のエントリの例（続き）

entPhysicalTable（続き）
entPhysicalEntry.entPhysicalIndex

entPhysicalEntry.24
 entPhysicalDescr　1gigethernet-1
 entPhysicalVendorType　CiscoModules.3.1.9.32.3
 entPhysicalContainedIn　2
 entPhysicalClass　module(9)
 entPhysicalParentRelPos　1

entPhysicalEntry.25
 entPhysicalDescr　Gigabit Ethernet MAC Controller
 entPhysicalVendorType　CiscoModules.3.1.10.109
 entPhysicalContainedIn　24
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　1
 entPhysicalName　GigE 1/0/0

entPhysicalEntry.26
 entPhysicalDescr　6cht3-1
 entPhysicalVendorType　CiscoModules.3.1.9.32.2
 entPhysicalContainedIn　3
 entPhysicalClass　module(9)
 entPhysicalParentRelPos　1

entPhysicalEntry.27
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　1
 entPhysicalName　Serial2/0/0

entPhysicalEntry.28
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　2
 entPhysicalName　Serial2/0/1

entPhysicalEntry.29
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　3
 entPhysicalName　Serial2/0/2

entPhysicalEntry.30
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　4
 entPhysicalName　Serial2/0/3

entPhysicalEntry.31
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　5
 entPhysicalName　Serial2/0/4

entPhysicalEntry.32
 entPhysicalDescr　PMC FREEM, PMC S/UNI...
 entPhysicalVendorType　CiscoModules.3.1.10.20
 entPhysicalContainedIn　26
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　6
 entPhysicalName　Serial2/0/5

entPhysicalEntry.33
 entPhysicalDescr　1oc12pos-1
 entPhysicalVendorType　CiscoModules.3.1.9.32.1
 entPhysicalContainedIn　4
 entPhysicalClass　module(9)
 entPhysicalParentRelPos　1

entPhysicalEntry.34
 entPhysicalDescr　Skystone 4302 Sonet Framer
 entPhysicalVendorType　CiscoModules.3.1.10.52
 entPhysicalContainedIn　33
 entPhysicalClass　port(10)
 entPhysicalParentRelPos　1
 entPhysicalName　POS3/0/0

76275

図 A-3 に、このコンフィギュレーションにおけるすべてのラインカードおよびラインカード ポートに関する entPhysicalTable のエントリを示します。

図 A-3　ラインカードおよびラインカード ポートに関する entPhysicalTable のエントリ

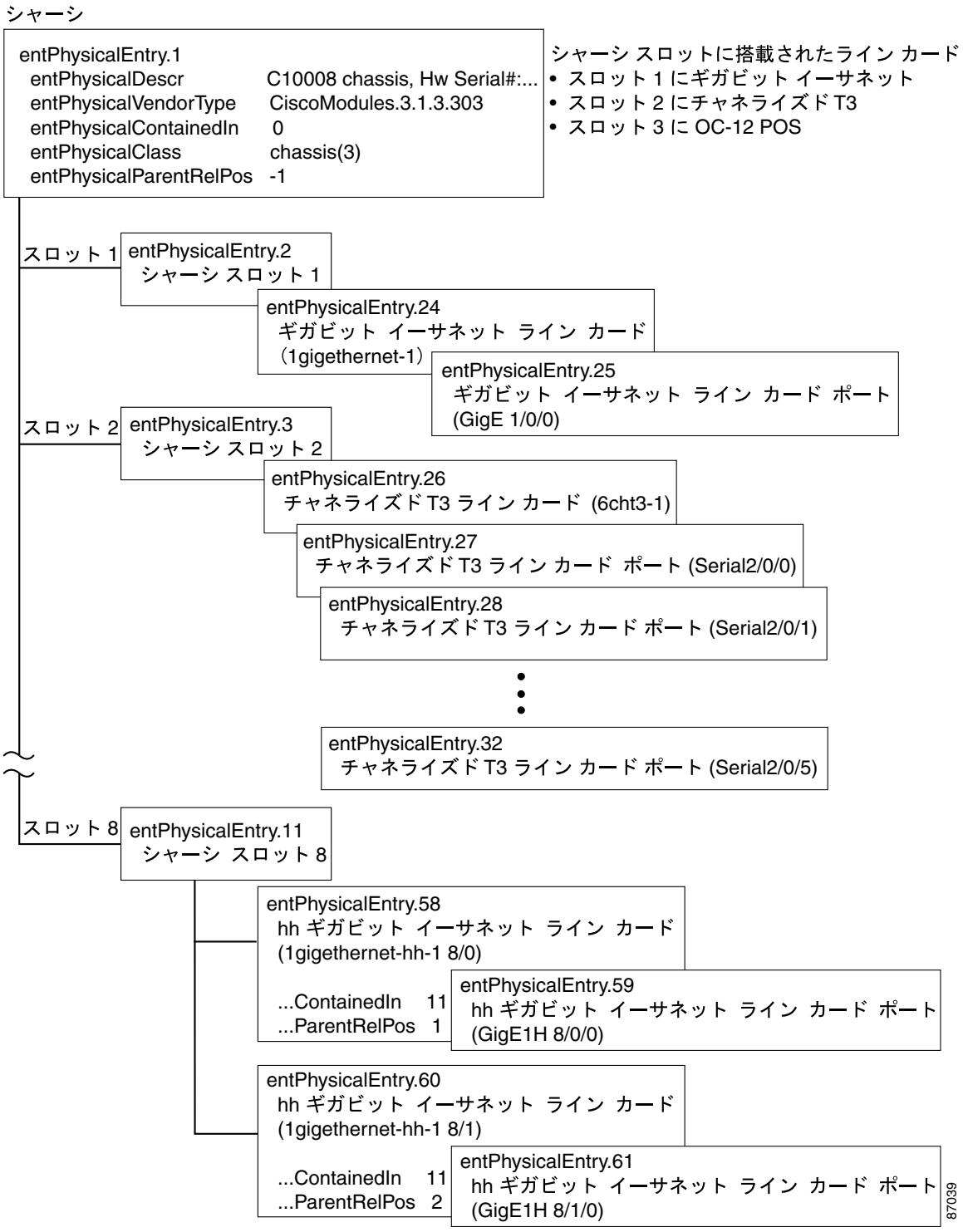
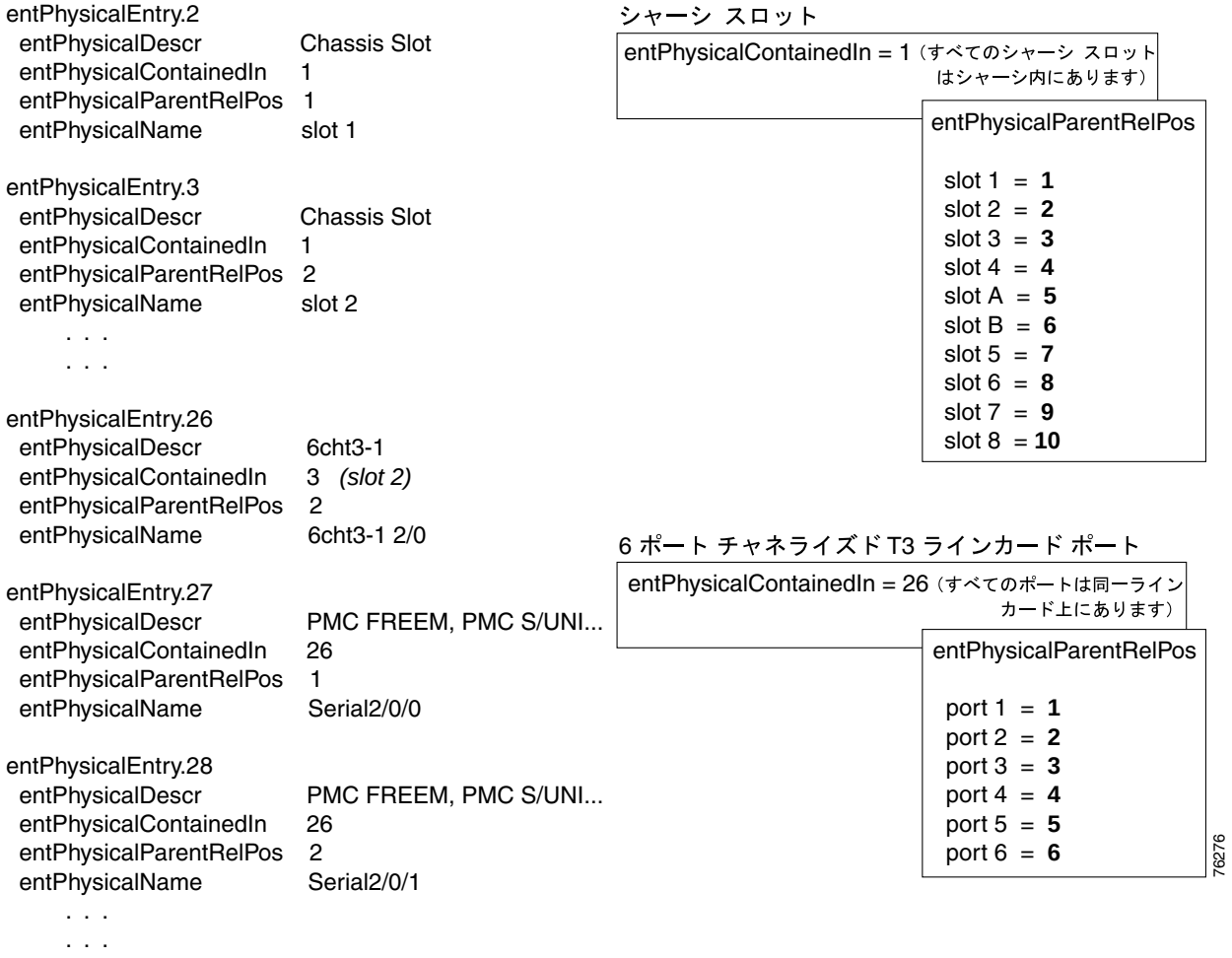


図 A-4 に、entPhysicalParentRelPos が親オブジェクト内での同タイプのエンティティの位置を表す仕組みを示します。図の左側の entPhysicalTable のエントリには、関連するフィールドのみが示されています。

図 A-4　　シャーシ スロットの entPhysicalParentRelPos 値



このコンフィギュレーション例について、次の点に注意してください。

- すべてのシャーシ スロットおよびラインカード ポートは、同じ entPhysicalContainedIn 値を持ちます。
  - シャーシ スロットでは、entPhysicalContainedIn = 1（シャーシの entPhysicalIndex）
  - ラインカード ポートでは、entPhysicalContainedIn =26（ラインカードの entPhysicalIndex）
- 各シャーシ スロットおよび各ラインカード ポートは、親オブジェクト内での相対的位置を示すために別々の entPhysicalParentRelPos を持ちます。
- 6 ポート チャネライズド T3 ラインカードは、シャーシのスロット 2 に挿入されています。

物理ポートの ifIndex 値の判別

ENTITY-MIB の **entAliasMappingIdentifier** は、物理ポートの entPhysicalIndex をそれに対応する IF-MIB の ifTable 上の ifIndex 値にマッピングすることによって、そのポートをインターフェイスにマッピングします。たとえば、次の例は、entPhysicalIndex が 35 の物理ポートが ifIndex 値が 4 のインターフェイスに対応することを示しています（適用できる MIB 値の詳細については、MIB を参照してください）。

```
entAliasMappingIdentifer.35.0 = ifIndex.4
```

ルータ資産のタギング

CISCO-ENTITY-ASSET-MIB の **ceAssetTag** オブジェクトを使用して、一意の不揮発性の識別子（タグ）を、追跡するあらゆるラインカードまたは Performance Routing Engine（PRE; パフォーマンス ルーティング エンジン）に対して割り当てることができます。PRE またはラインカードが他のシャーシに取り付けられた場合でも、タグはこの PRE またはラインカードに割り当てられたままとなります。

たとえば、次の ceAssetTable のエントリは、シリアル番号が CAB0430AXEU であるギガビット イーサネット ラインカードに、pre-1-1ge の資産タグが割り当てられていることを示しています。ラインカードがこのシャーシから取り外され他のシャーシに取り付けられた場合でも、その ceAssetTag は pre-1-1ge のままです。

```
ceSerialNumber = CAB0430AXEU
ceOrderablePartNumber = ESR-1GE
ceAssetTag = pre-1-1ge
  .  .  .
```

FRU ステータスの監視および設定

CISCO-ENTITY-FRU-CONTROL-MIB の cefcModuleTable 上のオブジェクトを調べて、FRU（電源装置やラインカードなど）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 の管理ステート。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 — 現在の FRU の動作ステート。

（注）　現在のところ、CISCO-ENTITY-FRU-CONTROL-MIB はラインカードのみをサポートしています。その他の MIB に関する制約事項については、「CISCO-ENTITY-FRU-CONTROL-MIB」（p.3-15）を参照してください。

entPhysicalIndex が 24 のギガビット イーサネット ラインカードに関する cefcModuleTable のエントリの例を示します。

```
cefcModuleEntry.24
  cefcModuleAdminStatus = enabled(1)
  cefcModuleOperStatus = ok(2)
  cefcModuleResetReason = manual reset(5)
  cefcModuleStatusLastChangeTime = 7714
```

Cisco 10000 シリーズ ESR が、FRU ステータスの変化を表すトラップを生成する手順については、「FRU ステータスの変更」（p.A-11）を参照してください。

SNMP トラップの生成

ここでは、Cisco 10000 シリーズ ESR 上のイベントおよび状態に応じて生成される SNMP トラップについて説明し、どのホストがトラップを受信するかを特定する方法について説明します。

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）を指定することができます。CLI に関する手順については、「トラップのイネーブル化」（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持ちます。snmpTargetAddrTable のホストがこの snmpNotifyTag 値と同じタグ値を持つ場合は、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 — これらのテーブル上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 と同じ場合には、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 通知を生成する際に使用する SNMP パラメータです。

該当する MIB 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 または mplsLdpSessionDown トラップを生成するためには、MPLS-LDP-MIB オブジェクト mplsLdpSessionUpDownTrapEnable が enabled（1）に設定されている必要があります。

設定の変更

エンティティ トラップがイネーブルの場合、Cisco 10000 シリーズ ESR は以下のいずれかのテーブル内の情報が変更されたとき（ESR 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

設定を変更すると必ず entConfigChange トラップが生成されるように Cisco 10000 シリーズ ESR 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ためには、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
```

FRU ステータスの変更

FRU トラップがイネーブルの場合、Cisco 10000 シリーズ ESR は FRU ステータスの変更に応じて、次のトラップを生成します。これらのトラップの詳細については、CISCO-ENTITY-FRU-CONTROL-MIB を参照してください。

- cefcModuleStatusChange — FRU の変更の動作ステータス（cefcModuleOperStatus）。
- cefcFRUInserted — FRU は、シャーシに挿入されています。このトラップは、FRU とその挿入先コンテナの 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 イベントに関するトラップを生成するように Cisco 10000 シリーズ ESR を設定するためには、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ためには、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
```

（注）　SNMP を使用して、FRU トラップをイネーブルにすることもできます。そのためには、cefcMIBEnableStatusNotification を true（1）に設定します。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。

アラーム

デフォルトでは、アラームがアサートまたはクリアされたときに SNMP がトラップおよび Syslog メッセージを生成します。ただし次の手順を実行すれば、トラップおよび Syslog メッセージが生成されるアラームのタイプを制御することも、トラップおよびメッセージを完全にディセーブルにすることもできます。

アラームに関するトラップのイネーブル化

Cisco 10000 シリーズ ESR のアラームに関する SNMP トラップをイネーブルまたはディセーブルにするには、次のいずれかを実行します。デフォルトでは、アラームがアサートまたはクリアされたときに SNMP がトラップを生成します。

- CLI で、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ためには、コマンドの **no** 形式を使用します。

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

- SNMP を使用して、次の CISCO-ENTITY-ALARM-MIB オブジェクトを設定します。

  ceAlarmNotifiesEnable — SNMP トラップが生成される原因となるアラームの重大度（critical[1]、major[2]、minor[3]、または info[4]）。トラップをディセーブルにするには ceAlarmNotifiesEnable を 0 に設定します。

  たとえば、major アラームおよび critical アラームに関するトラップを生成するには、ceAlarmNotifiesEnable を major に設定します。また、minor、major、および critical アラームに関するトラップを生成するには、ceAlarmNotifiesEnable を minor に設定します。アラームの重大度については、「アラームの重大度の説明」（p.A-15）を参照してください。

（注）　CISCO-ENTITY-ALARM-MIB は、ENTITY-MIB である entPhysicalTable で定義された物理エンティティのアラームのみを監視します。論理エンティティ（チャネライズド インターフェイスなど）のアラームは、Cisco IOS ソフトウェアが Syslog を介して監視します。

アラームの Syslog メッセージのイネーブル化

デフォルトでは、Cisco 10000 シリーズ ESR はアラームがアサートまたはクリアされるたびに Syslog メッセージを記録します。ただし、次の CISCO-SYSLOG-MIB オブジェクトを使用して、Syslog メッセージが生成されるアラームのタイプを設定することができます。

- ceAlarmSyslogEnable — Syslog メッセージを生成するアラームの重大度（critical[1]、major[2]、minor[3]、または info[4]）。アラームの重大度については、「アラームの重大度の説明」（p.A-15）を参照してください。これらの Syslog メッセージをディセーブルにするには、ceAlarmSyslogEnable を 0 に設定します。

また、Syslog メッセージがアラームに応じて記録されたときは、次の CISCO-SYSLOG-MIB オブジェクトが SNMP 通知を提供します。

- clogNotificationsEnabled — Syslog メッセージが記録されたときに通知を生成するかどうかを指定します。通知をイネーブルにするにはこのオブジェクトを true（1）に設定し、ディセーブルにするには false（2）に設定します。
- clogMessageGenerated — Syslog メッセージが生成されたときに送信される SNMP 通知。

環境状態

CISCO-ENVMON-MIB は、Cisco 10000 シリーズ ESR の環境センサで検出された状態に関して警告を発するために、次のトラップを生成します。

- ciscoEnvMonTemperatureNotification — 温度がクリティカルなしきい値を超えていることを示すために送信されます。ciscoEnvMonEnableTemperatureNotification オブジェクトを使用して、これらの通知をイネーブルまたはディセーブルにします。
- ciscoEnvMonShutdownNotification — ESR がシャットダウンしようとしていることを示すために送信されます。ciscoEnvMonEnableShutdownNotification オブジェクトを使用して、これらの通知をイネーブルまたはディセーブルにします。

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を生成するように Cisco 10000 シリーズ ESR を設定するためには、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ためには、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
```

（注）　SNMP を使用して、FRU トラップをイネーブルにすることもできます。そのためには、cefcMIBEnableStatusNotification を true（1）に設定します。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。

インターフェイスのリンク アップ / リンク ダウン ステート

インターフェイスおよびサブインターフェイスのリンク アップおよびリンク ダウン トラップをイネーブルまたはディセーブルするには、IF-MIB の ifLinkUpDownTrapEnable オブジェクトを enabled（1）または disabled（2）に設定します。デフォルトでは、これらのトラップはインターフェイス スタックの最下部レイヤに関してのみイネーブルになっています（つまり、インターフェイスの最下部レイヤがアップまたはダウンした時にのみトラップが生成されます）。

アラームの監視

アラームは、SONET パス上の信号損失やチャネライズド T3 ラインカード上の T1 インターフェイスのアクティブ化など、Cisco 10000 シリーズ ESR の状態を示します。アラームは、シャーシ、PRE、ラインカード、インターフェイスなどのエンティティのステータスに関するフィードバックを提供します。また、アラームおよびエラー メッセージは、ネットワーク パフォーマンスを低下させる可能性がある状態をユーザに知らせます。アラームはイベントではないことに注意してください。

各エンティティには一連のアラームがあり、エンティティで発生する可能性のある状態を定義しています。アラームは、次の情報を提供します。

- アラーム タイプ — アラームを識別する一意のコード
- 重大度 — アラームの原因となる状態の重大度
- 説明 — アラームの原因となった状態に関する情報

アラームのステートは、アラームの原因となった状態の現在のステートを示しています。

- Asserted — 現在もその状態が存在しています。
- Cleared — 状態は解消されました。

SNMP は、次のトラップを使用して現在のアラーム ステートを表します。

- ceAlarmAsserted — アラームの原因となった状態はまだ存在しています。
- ceAlarmCleared — アラームの原因となった状態は解消されました。

デフォルトでは、アラームがアサートまたはクリアされるたびに SNMP トラップおよび Syslog メッセージが生成されます。ただし、トラップおよび Syslog メッセージが生成されるアラームの重大度を設定するか、またはトラップおよびメッセージを完全にディセーブルにすることもできます（「アラーム」 [p.A-12] を参照してください）。

Cisco 10000 シリーズ ESR は、次を使用してアラームを監視します。

- CISCO-ENTITY-ALARM-MIB — ENTITY-MIB の entPhysicalTable で定義された、物理エンティティのアラーム。また、CISCO-SYSLOG-MIB はアラームに応じて Syslog メッセージの生成を制御します。
- Cisco IOS アラーム サブシステム — 物理エンティティのアラーム
- オメガ アラーム サブシステム — 論理エンティティとインターフェイスのアラーム（チャネライズド T3 ラインカード上の T1 インターフェイスなど）

目的および利点

Cisco 10000 シリーズ ESR のアラーム モニタ機能には、次のような利点があります。

- アラームの統計情報およびカウントを追跡します。
- アラームがアサートおよびクリアされた場合に監視します。
- アラームの履歴情報を入手します。
- アラームに応じて SNMP トラップを生成します。
- アラームに応じて Syslog メッセージを生成します。

アラームの監視に使用される MIB

- CISCO-ENTITY-ALARM-MIB — Cisco 10000 シリーズ ESR 上のアラームを監視することができます。この MIB には次のテーブルが含まれており、アラームの内容とアクティブなアラームの数を知ることができます。
  - ceAlarmDescrMapTable は、一意のアラーム インデックスをベンダー タイプにマッピングします。

- ceAlarmDescrTable は、ベンダー タイプおよびアラーム タイプごとにアラームを記述します。
- ceAlarmTable は、現在エンティティがアサートしているアラームを含む各エンティティのアラーム制御情報およびステータス情報を提供します。
- ceAlarmHistTable には、Cisco 10000 シリーズ ESR で発生したアラームの履歴が含まれています。

アラームに関する SNMP トラップおよび Syslog メッセージを生成する手順については、「アラーム」（p.A-12）を参照してください。

（注）　CISCO-ENTITY-ALARM-MIB は、ENTITY-MIB である entPhysicalTable で定義された物理エンティティのアラームのみを監視します。論理エンティティ（チャネライズド インターフェイスなど）のアラームは、Cisco IOS ソフトウェアが Syslog を介して監視します。

- CISCO-SYSLOG-MIB — SNMP を介して Syslog メッセージを監視するオブジェクトが含まれています。

アクティブなアラームの確認

Cisco 10000 シリーズ ESR 上のすべてのアクティブなアラームについて確認するためには、CLI から次のコマンドを入力します（*severity* は、表示されるアラームのレベルです）。ESR は、重大度（severity）がこのレベルおよびそれより高レベルのアクティブなアラームを、すべて表示します。*severity* を特定しない場合は、すべてのアクティブなアラームが表示されます。

```
Router(config)# show facility-alarm status [ severity ]
```

アラームの重大度の説明

次の一覧では、アラームの重大度、およびそれぞれの重大度が表す状態のタイプを説明しています。

- critical（1）— サービスに影響する重大な状態であり、即時に対処する必要があります。
- major（2）— サービス中断を引き起こすハードウェアまたはソフトウェア状態や、Cisco 10000 シリーズ ESR の動作上重要なハードウェアで発生したハードウェアまたはソフトウェア状態。重大度は critical アラームよりも低くなりますが、major アラームもすみやかに対処する必要があります。
- minor（3）— サービスに影響しない、または重要ではないハードウェア上で発生した状態または問題。
- info（4）— イベントに関する有益な情報メッセージ、または問題を引き起こす危険性のある状態の通知。

PXF 利用率の監視

ここでは、SNMP を使って Cisco 10000 シリーズ ESR 上で Parallel Express Forwarding Network Processor（PXF; パラレルエクスプレス フォワーディング ネットワーク プロセッサ）利用率を監視する方法を説明します。具体的な方法は次のとおりです。

- PXF 利用率と効率の確認
- PXF パフォーマンスのしきい値と再起動の監視

目的および利点

CISCO-ENTITY-PFE-MIB は Packet Forwarding Engine（PFE; パケット フォワーディング エンジン）にパフォーマンス情報への SNMP アクセスを提供します。PFE により特定の IP 機能が高速化され、ネットワーク パフォーマンスが向上します。MIB には、PFE 利用率と効率を監視するためのオブジェクトが含まれています。Cisco 10000 シリーズ ESR では、PFE は PXF であり、PRE の一部です。

この MIB には次のような利点があります。

- 次の CLI コマンドで PXF 利用率と効率を要約します。

```
show hardware pxf cpu context
```

- パフォーマンスの傾向に関する情報を、1 分ごと、5 分ごと、および 15 分ごとに提供します。
- 過去 24 時間の PXF 利用率と効率の履歴を保持します。
- ユーザ設定可能な利用率と効率のしきい値に対する PXF パフォーマンスを測定し、しきい値を超過した場合や PXF が再起動された場合はイベントを生成します。

PXF 利用率の監視に使用する MIB

- CISCO-ENTITY-PFE-MIB

PXF 利用率と効率の確認

CISCO-ENTITY-PFE-MIB 利用率と効率オブジェクトを使って、PXF パフォーマンスを測定します。

- PXF *利用率*とは、現在、処理に使用されている PXF の割合をパーセントで表したものです。PXF の処理が増えると、利用率も 0 から 100 パーセントにまで上がります。
- PXF *効率*では、PXF の実行状態が測定されます。値が大きいほど PXF 効率が高くなります。通常の稼働条件での一般的な PXF 効率は 100 パーセントです。効率が下がるとこの値も下がります。

Cisco 10000 シリーズ ESR の PXF 利用率と効率は、次の MIB テーブルの情報で確認できます。

- cePfePerfCurrentTable — 利用率および効率（パーセント表示）：現在、1 分 および 5 分
- cePfePerfIntervalTable — 過去 24 時間のパフォーマンス統計を 15 分間隔で表示。過去 24 時間以内に PXF が実行されていない場合でも、テーブルには 96 の測定間隔が表示されます。

  15 分間隔の開始時刻は最後に PXF が開始または再開された時刻によって異なり、実時間の 15 分ごとの区切り（たとえば 10:45 や 11:15）に対応しているわけではありません。たとえば PXF が 10:20 に開始すると、次は 10:35、その次は 10:50 のように、15 分おきに PXF が開始されます。
- cePfePerfTotalTable — 過去 24 時間の利用率と効率。

PXF パフォーマンスのしきい値と再起動の監視

CISCO-ENTITY-PFE-MIB を使って PXF 利用率と効率のしきい値を設定したり、設定したしきい値に対する PXF パフォーマンスを監視したりすることができます。SNMP は PXF パフォーマンス（ceCpfePerfCurrentTable）をしきい値（cePfePerfConfigTable）と比較して、PXF 利用率または効率がしきい値に達したりしきい値を超過した場合にイベントを生成します。イベントをイベント履歴テーブル（cePfeHistTable）のログに記録し、SNMP 通知を生成することができます。両方のアクションを実行するか、もしくはどちらのアクションも実行しないでください。また、PXF イベントは、PXF が再起動するごとに生成されます。

たとえば、1 分ごとに利用率を測定する PXF（cePfePerfCurrent1MinUtilization）がしきい値（cePfePerfThld1MinUtilization）に達したりしきい値を超過した場合、SNMP は thld1MinUtilizationEvent を生成します。イベントの説明については、MIB オブジェクト HistEventType を参照してください。

PXF しきい値を設定してから監視することによって PXF 利用率と効率を追跡するには、次の手順を実行します。

ステップ 1　cePfePerfConfigTable で、PXF 利用率と効率に使用できるしきい値を定義します。

ステップ 2　cePfeHistNotifiesEnable に次のいずれかの値を設定し、しきい値を超過した場合や PXF が再起動した場合の SNMP のアクションを指定します。

- none（1）— SNMP は何のアクションも実行しません。これがデフォルトです。
- log（2）— cePfeHistTable にエントリを作成します。
- notify（3）— SNMP 通知を送信します。
- logAndNotify（4）— cePfeHistTable エントリを作成し、SNMP 通知を送信します。

ステップ 3　cePfeHistTable にイベントを記録するには、cePfeHistTableSize を使ってテーブルで使用できるエントリの最大数を指定します。テーブルが満杯になったら、新しいイベントはテーブルの最も古いエントリを上書きします。

ステップ 4　cePfeHistNotifiesEnable を log（2）または logAndNotify（4）に設定すると、超過したしきい値と PXF 再起動を cePfeHistTable で確認できます。

ラインカードの事前プロビジョニング

ここでは、ラインカードの事前プロビジョニング プロセスおよび SNMP データへの影響について説明します。この事前プロビジョニング機能では、実際にラインカードをシャーシに挿入する前に、特定のラインカード タイプに対応するようラインカード スロットを事前設定します。システムは、システムの実行コンフィギュレーション ファイルにラインカードの基本設定を追加し、実際にシャーシにラインカードが挿入されたときにこの設定を適用します。

（注）　SNMP を使用してラインカード スロットを事前プロビジョニングすることはできません。CLI の **card** コマンドを使用する必要があります。詳しい設定については、CCO の Cisco 10000 Series ESR New Features のマニュアル リンクにある『*Cisco 10000 Series ESR Line Card Slot Preprovisioning*』機能の説明を参照してください。

事前プロビジョニングしたラインカードの取り付け

ラインカードが Cisco 10000 のシャーシに挿入されたときに次のような動作が発生します。

- 挿入したラインカードが、スロットに事前プロビジョニングしたラインカードのタイプと一致する場合、システムは事前プロビジョニングした設定を適用します。
- ラインカード スロットが事前プロビジョニングされていない場合、システムはラインカードに基本設定を適用し、この設定を実行コンフィギュレーション ファイルに追加します。
- ラインカード スロットに事前プロビジョニングしたラインカード タイプと異なるラインカードが挿入された場合、システムは、事前プロビジョニングした設定（実行コンフィギュレーション ファイル内）を実際に挿入されたラインカードの基本設定に置き換えます。

スロットが使用するよう設定されているカード タイプを調べるには、**show running-config** コマンドを入力します。

MIB への影響

次の MIB テーブルの情報が影響を受けるのは、事前プロビジョニングされたラインカードが Cisco 10000 のシャーシに挿入された場合、またはラインカードの動作に影響する CLI コマンドを入力した場合です。

- ENTITY-MIB（entPhysicalTable および entAliasMappingTable）
- CISCO-ENTITY-ASSET-MIB（ceAssetTable）
- CISCO-ENTITY-FRU-CONTROL-MIB（cefcModuleTable）

ラインカードの交換 — MIB ステート特性

Cisco 10000 シリーズ ESR のシャーシに搭載されているラインカードを交換すると、MIB は次のような影響を受けます。

- 別のタイプのラインカードと交換する場合（たとえば、ギガビット イーサネット ラインカードをチャネライズド T3 ラインカードと交換する場合）、元のラインカードの情報が MIB から削除されます。
- *同じ*タイプの別のラインカードと交換する場合、MIB は元のラインカードの設定を保持します。このため、設定情報を失うことなくラインカードが交換できます。たとえば、サブインターフェイスが 50 あるギガビット イーサネット ラインカードを交換する場合、MIB にはそのラインカードとサブインターフェイスの情報が保持されます。MIB にある元のラインカード情報を交換する方法は、次の手順を参照してください。

ラインカードを同じタイプの別のラインカードと交換し、MIB から元のラインカード情報を削除するには、次の手順を実行します。

ステップ 1　次の CLI コマンドを発行してラインカードをシャットダウンします（*slot_number* は 1 ～ 8 です）。

```
hw-module slot slot_number shutdown
```

ステップ 2　そのまま 30 秒待ち、ラインカード異常 LED が点灯するのを確認します。

ステップ 3　シャーシからラインカードを取り外します。

（注）　このスロットにすぐに別のラインカードを取り付ける予定がない場合は、**no card** コマンドを発行します（ステップ 4 を参照）。

ステップ 4　次の CLI コマンドを発行し、Cisco 10000 シリーズ ESR コンフィギュレーションおよび MIB からラインカード情報を削除します。たとえば、コマンド **no card 5/0** を発行すると、スロット 5 のラインカードの設定情報が削除されます。

```
no card slot/subslot
```

ステップ 5　（任意）MIB からラインカード設定情報が削除されたことを確認するには、MIB の内容を表示します。

ステップ 6　シャーシ スロットに新しいラインカードを挿入します。

ステップ 7　次の CLI コマンドを発行し、新しく取り付けたライン ードをアクティブにします。

```
no hw-module slot slot_number shutdown
```

バルク ファイル取得の実行

ここでは、SNMP を使用して Cisco 10000 シリーズ ESR から大量のデータをバルク取得する方法について説明します。この機能を使用して、SNMP エージェントとマネージャの間でさまざまな情報（QoS 統計、インターフェイス統計、entPhysicalTable エントリなど）を転送することができます。

目的および利点

従来は、Cisco 10000 シリーズ ESR から大量のデータを検索するには、管理アプリケーションから SNMP **get-next** または **get-bulk** 要求を何度も発行する必要がありました。

SNMP のバルク ファイル検索機能を使用すると、このプロセスが簡素化され、パフォーマンスの向上につながります。バルク ファイル検索は、次の手順で行います。

1. バルク ファイルの特性を定義する。
2. バルク ファイルに含めるデータを指定する。
3. バルク ファイルを作成し、転送するデータを書き込む。
4. FTP（ファイル転送プロトコル）ユーティリティを使用して、バルク ファイルをルータから他のシステムにコピーする。

バルク ファイル検索は、次のいずれかの方法で行います。

- ホストから SNMP **set** および **get** 要求を発行する（コマンドの例は「SNMP コマンド」[p.A-21]を参照）。
- SNMP **set** および **get** 要求を発行する管理アプリケーションを作成する。

Cisco 10000 シリーズ ESR の MIB 拡張機能には、バルク取得処理機能により、ルータの ifTable を取得する Java アプレットも含まれています（「Java アプレット」[p.A-23]を参照）。

バルク ファイルの取得に使用する MIB

- CISCO-BULK-FILE-MIB
- CISCO-FTP-CLIENT-MIB

バルク ファイル取得の手順

ここでは、バルク ファイル取得の実行手順を説明します。このプロセスの例は、「SNMP コマンド」（p.A-21）および「Java アプレット」（p.A-23）を参照してください。MIB オブジェクト、その特性、および有効な値についての詳細は、MIB を参照してください。

ステップ 1　CISCO-BULK-FILE-MIB の cbfDefineFileTable に行を 1 つ作成し、バルク ファイルの特性を定義します。

a. この行に割り当てる一意のインデックスを決定します。

b. cbfDefineFileTable オブジェクトを設定します。

cbfDefineFileEntryStatus = createAndGo(4) or createAndWait(5)
cbfDefineFileName = *bulk_file_name*
cbfDefineFileStorage = ephemeral(1)
cbfDefineFileFormat = bulkASCII(3)

ステップ 2　cbfDefineObjectTable に行を 1 つ作成し、バルク ファイルに含めるデータを定義します。

a. この行に割り当てる一意のインデックスを決定します。

b. cbfDefineObjectTable オブジェクトを設定します。

cbfDefineFileObjectStatus = createAndGo(4) or createAndWait(5)
cbfDefineObjectID = *the OID of the object instance or table column*
cbfDefineObjectClass = object(1) or lexicaltable(2)

c. （任意）バルク ファイルに特定のテーブル行を含めるには、次の MIB オブジェクトを設定します。このオプションを使用するには、cbfDefineObjectClass = lexicaltable（2）を設定する必要があります。

cbfDefineObjectTableInstance = *starting table row*
cbfDefineObjectNumEntries = *number of table rows to include*

たとえば、ifTable の行 2 ～ 12 を含めるには、cbfDefineObjectTableInstance = ifTable.2 および cbfDefineObjectNumEntries = 10 と設定します。

ステップ 3　バルク ファイルを作成し、そのファイルにデータを書き込み、ファイルのステータスを確認します。

a. cbfDefineFileNow = create（3）を設定します。

b. バルク ファイルの cbfDefineFileIndex を使用して、cbfStatusFileState について **get-next** を実行します。

c. cbfStatusFileState = ready（2）になるまで、待ちます。

ステップ 4　CISCO-FTP-CLIENT-MIB の cfcRequestTable に行を 1 つ作成し、バルク ファイルを FTP サーバにコピーするように FTP を設定します。

a. この行に使用する一意のインデックスを決定します。

b. cfcRequestTable オブジェクトを設定します。

cbfRequestEntryStatus = createAndWait(5) or createAndGo(4)
cfcRequestOperation = putASCII(2)
cfcRequestLocalFile = *ルータ上のバルク ファイルの名前*
cfcRequestRemoteFile = *宛先 FTP サーバにバルク ファイルをコピーする名前（およびパス）*
cfcRequestServer = *IP アドレスまたは FTP サーバの完全記述名*
cfcRequestUser = *FTP サーバ用の有効なユーザ名*
cfcRequestPassword = *FTP ユーザ名のパスワード*

c. cfcRequestEntryStatus を active（1）に設定して行をアクティブにします。バルク ファイルの転送が開始されます。

d. cfcRequestResult をチェックして、FTP 動作の結果を確認します。

SNMP コマンド

図 A-5 に、バルク ファイル検索に使用する SNMP コマンドの例を示します。これらのコマンドは一例に過ぎません。実際に使用するコマンドは、例とは異なる場合があります。有効な値については、MIB を参照してください。この例で使用する数値は、「バルク ファイル取得の手順」（p.A-20）に示す手順に対応しています。各ステップについて表 A-1 で説明します。

（注）　この例では、SNMP EMANATE ツール（SNMP Research International 製）を利用しています。以下に示すコマンドで、*rtr_IP_addr* は、Cisco 10000 シリーズ ESR の IP アドレスです。

図 A-5　バルク ファイル検索に使用する SNMP コマンドの例

```
① setany -v2c rtr_IP_addr private cbfDefineFileEntryStatus.1 -i 4
  setany -v2c rtr_IP_addr private cbfDefineFileName.1 -D "QoSstats"
  setany -v2c rtr_IP_addr private cbfDefineFileStorage.1 -i 1
  setany -v2c rtr_IP_addr private cbfDefineFileFormat.1 -i 3

② setany -v2c rtr_IP_addr private cbfDefineObjectEntryStatus.1.3 -i 4
  setany -v2c rtr_IP_addr private cbfDefineObjectID.1.3 -d 1.3.6.1.2.1.4.20
  setany -v2c rtr_IP_addr private cbfDefineObjectClass.1.3 -i 2

③ setany -v2c rtr_IP_addr private cbfDefineFileNow.1 -i 3
  getone -v2c rtr_IP_addr private cbfStatusFileState.1.1

④ setany -v2c rtr_IP_addr private cfcRequestEntryStatus.1 -i 5
  setany -v2c rtr_IP_addr private cfcRequestOperation.1 -i 2
  setany -v2c rtr_IP_addr private cfcRequestLocalFile.1 -D "QoSstats"
  setany -v2c rtr_IP_addr private cfcRequestRemoteFile.1 -D "C10kQoS"
  setany -v2c rtr_IP_addr private cfcRequestServer.1 -D "stats.cisco.com"
  setany -v2c rtr_IP_addr private cfcRequestUser.1 -D "JoeSmith"
  setany -v2c rtr_IP_addr private cfcRequestPassword.1 -D "bluefi$h"
  setany -v2c rtr_IP_addr private cfcRequestEntryStatus.1 -i 1
```

69731

表 A-1　SNMP コマンド例の解説

ステップ 1	バルク ファイル QoSstats に対応する行を 1 つ作成し、この行を Active ステートにします。 この行に 1 というインデックスを割り当てます。このインデックスを使用して、行に対応するテーブル オブジェクト（たとえば、cbfDefineFileEntryStatus**.1**、cbfDefineFileName**.1**、cbfDefineFileStorage**.1** など）にアクセスします。 バルク ファイルが読み込まれるまで、ファイルに保存されているのはデータだけです。バルク ファイルのフォーマットは、人間が読める ASCII 形式です。
ステップ 2	インデックス **.1.3**（cbfDefine**File**Index および cbfDefine**Object**Index）で行を 1 つ作成します。 この MIB 設定によって、ipAddrTable のデータをバルク ファイルに含めることを指定します。
ステップ 3	バルク ファイルを作成し、cbfDefineFileIndex を使用してバルク ファイルが作成されたことを確認します。
ステップ 4	このコマンドによって、バルク ファイル（QoSstats）を stats.cisco.com サーバのファイル C10kQoS にコピーする FTP **put** 要求を設定します。デフォルトでは、指定したユーザのホーム ディレクトリにファイルがコピーされますが、別のディレクトリを指定することもできます。たとえば、cfcRequestRemoteFile = /C10Kstats/QoSstats と指定すると、ディレクトリ /C10Kstats にバルク ファイルがコピーされます。

SNMP コマンドを使用する際、次の点に注意してください。

- テーブルの行ごとに、その行のテーブル オブジェクトにアクセスするために使用する一意のインデックスを決定する必要があります。このインデックスは、テーブル内のすべての行で一意でなければなりません。行を作成する時点で、インデックスを定義する必要があります。
- テーブルに行を 1 つ作成するには、
  - テーブルの *xxxEntryStatus* オブジェクトに行のインデックスを追加します。
  - *xxxEntryStatus* = createAndGo（4）または createAndWait（5）に設定します。

  システムは指定された行を作成し、指定されたインデックスをその行に割り当てます。たとえば、次のコマンドを使用すると、cbfDefineFileTable に行が 1 つ作成され、その行は Inactive ステートに設定されます。この行には 1 というインデックスが割り当てられます。

  ```
  setany -v2c rtr_IP_addr private cbfDefineFileEntryStatus.1 -i 5
  ```

  このインデックス値を使用して、その行にある別の MIB オブジェクト（cbfDefineFileName.1、cbfDefineFileStorage.1 など）にアクセスします。

Java アプレット

Java アプレットを使用してルータの ifTable のバルク ファイル検索を実行するには、次の作業を行います。

1. Cisco 10000 シリーズ ESR が、（アプレットの実行に必要な）Java 2 プラットフォームをサポートするワークステーションに接続されていることを確認します。
2. シスコ FTP サイトの次の URL にアクセスします。

   ftp://ftp-eng.cisco.com/auto/ftp/omega/cheops
3. FTP サイトから次のファイルをワークステーションにコピーします。

   10kApplets.jar
4. ワークステーションで、次のコマンドを発行してアプレットを起動します（applet.BulkFileRetrieval は、JAR ファイル内のバルク ファイル検索クラスまたはプログラムを実行するようシステムに指示します）。

   ```
   java -cp <JAR_file_location> applet.BulkFileRetrieval
   ```
5. 表示されるウィンドウに、次の情報を入力します。
   - Cisco 10000 シリーズ ESR 上のイーサネット ポートの IP アドレス
   - SNMP バージョンおよびコミュニティ
   - バルク ファイルの転送先 FTP サーバの IP アドレス
   - サーバに対して有効なユーザ名およびパスワード
   - 指定したユーザのホーム ディレクトリ（バルク ファイルはこの場所にコピーされます）
6. **Retrieve BULK-FILE** をクリックして、バルク ファイル検索を開始します。

   システムは ifTable を ifTable–bulkFile<*MonthDay–HourMinSec*> という名前のバルク ファイル（たとえば、ifTable–bulkFileJan3–17hr9min16sec）にコピーし、次にこのバルク ファイルを転送先 FTP サーバにある指定ユーザのホーム ディレクトリにコピーします。
7. **BULK-FILE Data** タブをクリックして、バルク ファイルを表示します。

図 A-6 に、Java アプレットを使用するバルク ファイル検索の例を示します。一連の番号は、「バルク ファイル取得の手順」（p.A-20）の各ステップに対応しています。アプレットの各ステップについての説明は、表 A-2 を参照してください。

図 A-6　Java アプレットによるバルク ファイル検索

```
public int createCbfDefineFileRow(String bulkFileName) {

(1a)  int[] cbfDefineFileTableIndex = {getRandomNumber()};
      String indexValue = snmp.makeOIDFromArray(cbfDefineFileTableIndex);

(1b)  snmp.snmpSet_addVarbind("cbfDefineFileEntryStatus",
                              indexValue,
                              RowStatusEnum_createAndGo);

      snmp.snmpSet_addVarbind("cbfDefineFileName",
                              indexValue,
                              bulkFileName);

      snmp.snmpSet_addVarbind("cbfDefineFileStorage",
                              indexValue,
                              FileStorageEnum_ephemeral);

      snmp.snmpSet_addVarbind("cbfDefineFileFormat",
                              indexValue,
                              FileFormatEnum_bulkASCII);

      snmp.snmpSet_go();
}


public boolean createCbfDefineObjectRow(int bulkFileId,
                                        String objectClass,
                                        String objectId) {

(2a)  int[] cbfDefineObjectTableIndex = {bulkFileId, getRandomNumber()};
      String indexValue = snmp.makeOIDFromArray(cbfDefineObjectTableIndex);

(2b)  snmp.snmpSet_addVarbind("cbfDefineObjectEntryStatus",
                              indexValue,
                              RowStatusEnum_createAndGo);

      snmp.snmpSet_addVarbind("cbfDefineObjectClass", indexValue, objectClass);

      snmp.snmpSet_addVarbind("cbfDefineObjectID", indexValue, objectId);

      snmp.snmpSet_go();
      }


      public boolean startAndMonitorBulkFileCreation(int bulkFileId) {

(3a)        String indexValue = "." + bulkFileId;
            snmp.snmpSet_addVarbind("cbfDefineFileNow",
                                    indexValue,
                                    FileNowEnum_create);
            snmp.snmpSet_go();

(3b)        SnmpVarBind result = snmp.snmpGetNextObject("cbfStatusFileState", indexValue);
            String fileStateIndex = snmp.extractIndexValues(result);
            int fileStateValue = result.getValue()
```

69732

図 A-6　　Java アプレットによるバルク ファイル検索（続き）

```
3c  while (fileStateValue == FileStateEnum_running) {
            sleep(FileStatePollInterval);
            result = snmp.snmpGetObject("cbfStatusFileState", fileStateIndex);
            fileStateValue = result.getValue();
        }

        boolean returnResult = determineResult(fileStateValue);
        return returnResult;
    }


    public boolean startAndMonitorBulkFileTransfer(String bulkFileName,
        String     bulkFileType,
        String     ftpServerAddr,
        String     ftpServerUsername,
        String     ftpServerPassword) {

4a      int[] cfcRequestTableIndex = {getRandomNumber()};
        String indexValue = snmp.makeOIDFromArray(cfcRequestTableIndex);

4b      snmp.snmpSet_addVarbind("cfcRequestEntryStatus",
                                indexValue,
                                RowStatusEnum_createAndWait);

        snmp.snmpSet_addVarbind("cfcRequestOperation",
                                indexValue,
                                bulkFileType);

        snmp.snmpSet_addVarbind("cfcRequestLocalFile",
                                indexValue,
                                bulkFileName);

        snmp.snmpSet_addVarbind("cfcRequestRemoteFile",
                                indexValue,
                                bulkFileName);

        snmp.snmpSet_addVarbind("cfcRequestServer",
                                indexValue,
                                ftpServerAddr);

        snmp.snmpSet_addVarbind("cfcRequestUser",
                                indexValue,
                                bulkFileUsername);

        snmp.snmpSet_addVarbind("cfcRequestPassword",
                                indexValue,
                                ftpServerPassword);

        snmp. snmpSet_go();
```

69733

図 A-6　Java アプレットによるバルク ファイル検索（続き）

```
(4c) snmp.snmpSet_addVarbind("cfcRequestEntryStatus",
                                indexValue,
                                RowStatusEnum_active);

     returnStatus = snmp.snmpSet_go();

(4d) SnmpVarBind result = snmp.snmpGetObject("cfcRequestResult", indexValue);
     int requestResultState = result.getValue();
     int timeToCompletion = 0;
     while (requestResultState == cfcRequestResult_pending ||
                    timeToCompletion < BulkFileTransferTimeout) {
               sleep(cfcRequestStatePollInterval);
               timeToCompletion+=cfcRequestStatePollInterval;
               result = snmp.snmpGetObject("cfcRequestResult", indexValue);
               requestResultState = result.getValue();
     }

     boolean returnResult = determineResult(fileStateValue);
     return returnResult;
}
```

69734

表 A-2　applet.BulkFileRetrieval の説明

ステップ 1a	インデックスとして使用する乱数を発生させます。
ステップ 1b	次のように MIB オブジェクトを設定します。 cbfDefineFileStorage = ephemeral(1) cbfDefineFileFormat = bulkASCII(3)
ステップ 2a	インデックスとして使用する乱数を発生させます。
ステップ 3b	cbfDefineFileIndex を使用して cbfStatusFileState にアクセスします。
ステップ 4c	ステートが running（1）から ready（2）に変わるまで、cbfStatusFileState をポーリングします。
ステップ 5d	バルク ファイル転送ペンディングが終わるまで、cfcRequestResult をポーリングします。

QoS の監視

ここでは、Cisco 10000 シリーズ ESR 上で SNMP を使用して QoS（Quality of Service）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。具体的な内容は次のとおりです。

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** コマンドを発行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 Cisco 10000 シリーズ ESR 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 に使用する MIB

- CISCO-CLASS-BASED-QOS-MIB

QoS の設定

QoS を設定するには、Cisco 10000 シリーズ ESR の CLI を使用します。詳しい設定手順については、『*Cisco 10000 Series ESR 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 Cisco 10000 シリーズ ESR 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 でアクセスすることはできます。

QoS インデックス

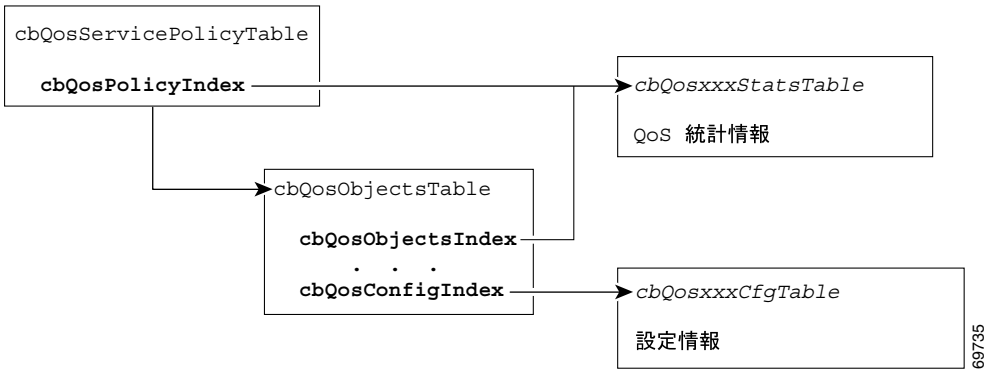
QoS 設定情報および統計情報へのアクセスに使用するインデックスは、次のとおりです。

- 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 ステートメント、フィーチャ 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。同じ設定を持つ QoS オブジェクトは、同じ cbQosConfigIndex を共有します。

- cbQosREDValue — Weighted Random Early Detection（WRED; 重み付けランダム早期検出）アクションの IP precedence または IP Differentiated Services Code Point（DSCP）。各 Randam Early Detection（RED; ランダム早期検出）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-7 に、これらのインデックスによって 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ます。

図 A-7　Cisco 10000 ESR 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、コンフィギュレーション テーブル（cbQos*xxx*CfgTable）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（cbQos*xxx*StatsTable）で QoS 機能の情報にアクセスします。

QoS 設定の例

ここでは、CISCO-CLASS-BASED-QOS-MIB テーブルに保存された QoS 設定の例を一連の図で示します。

- 図 A-8 は、他の図の基盤となっている QoS 設定例を示しています。
- 図 A-9 は、サービス ポリシーおよびオブジェクト テーブルを示しています。
- 図 A-10 は、ポリシー マップ、クラス マップ、およびポリシング アクションの設定情報を示しています。
- 図 A-11 は、ATM インターフェイスに関する RED クラスの設定を示しています。

この項で示す一連の図では、QoS オブジェクト別に情報が示されています。ただし、SNMP クエリ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、図 A-8 に示す設定に対応する QoS 情報の一部に過ぎません。

```
c10k# getmany -v3 10.86.0.94 test-user ciscoCBQosMIB

cbQosIfType.1047 = subInterface(2)
cbQosIfType.1052 = subInterface(2)
cbQosPolicyDirection.1047 = input(1)
cbQosPolicyDirection.1052 = output(2)
cbQosIfIndex.1047 = 36
cbQosIfIndex.1052 = 36
cbQosFrDLCI.1047 = 0
cbQosFrDLCI.1052 = 0
cbQosAtmVPI.1047 = 0
cbQosAtmVPI.1052 = 0
cbQosAtmVCI.1047 = 0
cbQosAtmVCI.1052 = 0
cbQosConfigIndex.1047.1047 = 1045
cbQosConfigIndex.1047.1048 = 1025
cbQosConfigIndex.1047.1050 = 1027
cbQosConfigIndex.1047.1051 = 1046
cbQosConfigIndex.1052.1052 = 1045
cbQosConfigIndex.1052.1053 = 1025
cbQosConfigIndex.1052.1055 = 1027
cbQosConfigIndex.1052.1056 = 1046
cbQosObjectsType.1047.1047 = policymap(1)
cbQosObjectsType.1047.1048 = classmap(2)
cbQosObjectsType.1047.1050 = matchStatement(3)
cbQosObjectsType.1047.1051 = police(7)
cbQosObjectsType.1052.1052 = policymap(1)
cbQosObjectsType.1052.1053 = classmap(2)
cbQosObjectsType.1052.1055 = matchStatement(3)
cbQosObjectsType.1052.1056 = police(7)
cbQosParentObjectsIndex.1047.1047 = 0
cbQosParentObjectsIndex.1047.1048 = 1047
cbQosParentObjectsIndex.1047.1050 = 1048
cbQosParentObjectsIndex.1047.1051 = 1048
cbQosParentObjectsIndex.1052.1052 = 0
cbQosParentObjectsIndex.1052.1053 = 1052
cbQosParentObjectsIndex.1052.1055 = 1053
cbQosParentObjectsIndex.1052.1056 = 1053
cbQosPolicyMapName.1045 = pm-1Meg
cbQosPolicyMapDesc.1045 =
cbQosCMName.1025 = class-default
cbQosCMDesc.1025 =
cbQosCMInfo.1025 = matchAny(3)
     . . .
```

図 A-8　GigabitEthernet QoS 設定 — CLI の show コマンド

```
c10k# show class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-any cm1 (id2)
  Description: class map #1
   Match ip  dscp 48

c10k# show policy-map
  Policy Map pm1
    Class cm1
      shape 1200000
      random-detect dscp-based
      random-detect dscp 32 202 8000 20
      random-detect dscp 44 200 6000 22
      random-detect dscp 61 201 7000 22
  Policy Map pm-1Meg
    Class class-default
      police 1000000 8000 8000 conform-action transmit exceed-action drop

c10k# show policy-map interface
 GigabitEthernet1/0/0.1

  Service-policy input: pm-1Meg (1057)

    Class-map: class-default (match-any) (1058/0)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any  (1060)
        0 packets, 0 bytes
        5 minute rate 0 bps
      Police:
        1000000 bps, 8000 limit, 8000 extended limit
        conformed 0 packets, 0 bytes; action: transmit
        exceeded 0 packets, 0 bytes; action: drop

  Service-policy output: pm-1Meg (1062)

    Class-map: class-default (match-any) (1063/0)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any  (1065)
        0 packets, 0 bytes
        5 minute rate 0 bps
      Output queue: 0/8192; 0/0 packets/bytes output, 0 drops
      Police:
        1000000 bps, 8000 limit, 8000 extended limit
        conformed 0 packets,0 bytes; action: transmit
        exceeded 0 packets, 0 bytes; action: drop
```

69736

図 A-9　GigabitEthernet QoS — サービス ポリシーおよびオブジェクト テーブル

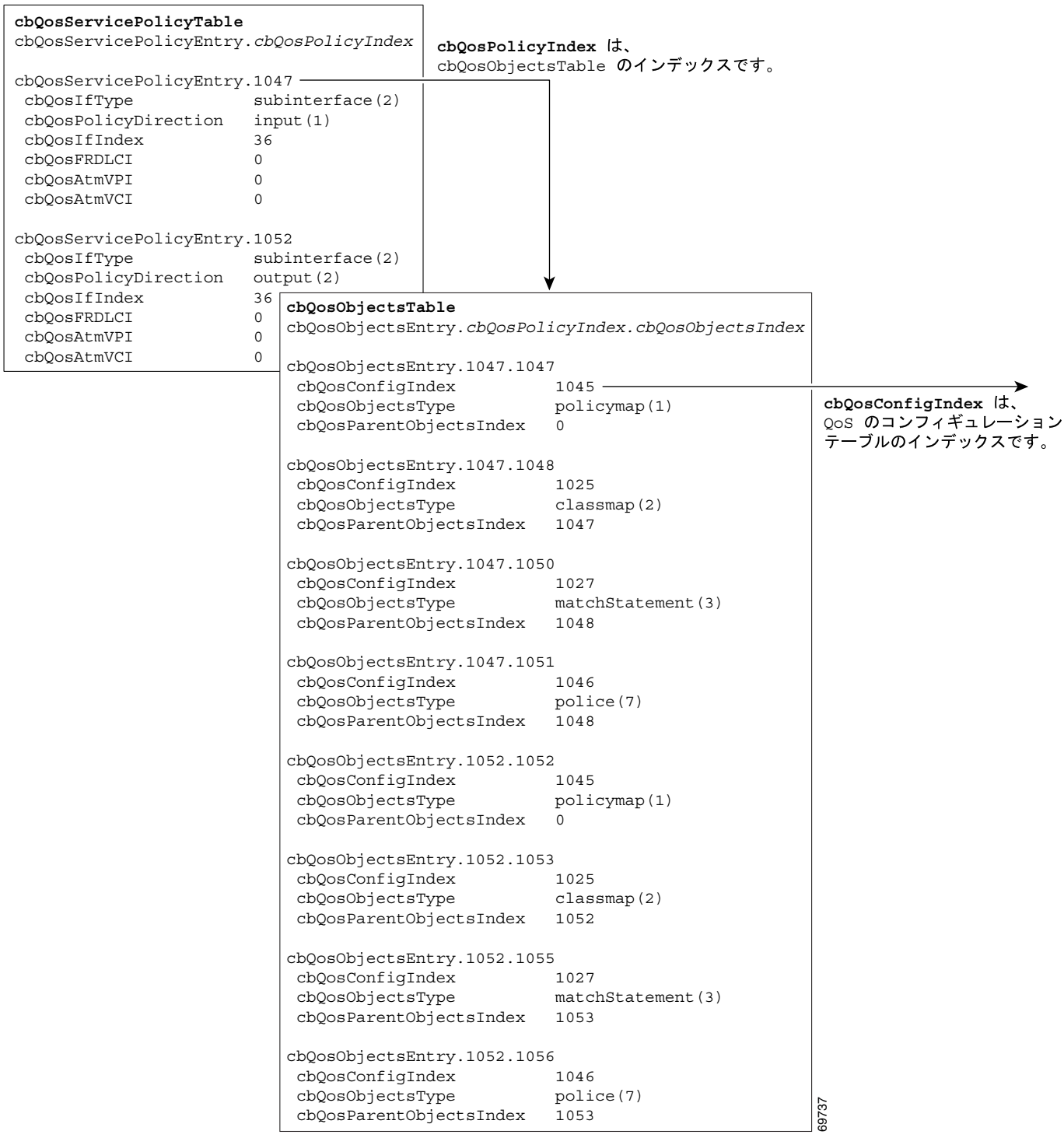


図 A-10　GigabitEthernet QoS ― ポリシー マップ、クラス マップ、およびポリシング アクションのコンフィギュレーション オブジェクト

```
c10k# show policy-map
        . . .
  Policy Map pm-1Meg
    Class class-default
      police 1000000 8000 8000 conform-action transmit
         exceed-action drop

c10k# show policy-map interface
 GigabitEthernet1/0/0.1

  Service-policy input: pm-1Meg (1057)
        . . .
  Service-policy output: pm-1Meg (1062)
        . . .

c10k# show class-map
 Class Map match-any class-default (id 0)
   Match any
        . . .

c10k# show policy-map
        . . .
  Policy Map pm-1Meg
    Class class-default
      police 1000000 8000 8000 conform-action transmit
         exceed-action drop
        . . .
```

cbQosPolicyMapCfgTable
cbQosPolicyMapCfgEntry.*cbQosConfigIndex*

cbQosPolicyMapCfgEntry.1045
 cbQosPolicyMapName　pm-1Meg
 cbQosPolicyMapDesc

cbQosMatchStmtCfgTable
cbQosMatchStmtCfgEntry.*cbQosConfigIndex*

cbQosMatchStmtCfgEntry.1027
cbQosMatchStmtName　Match any
cbQosMatchStmtInfo　none (1)

cbQosCMCfgTable
cbQosCMCfgEntry.*cbQosConfigIndex*

cbQosCMCfgEntry.1025
 cbQosCMName　class-default
 cbQosCMDesc
 cbQosCMInfo　matchAny(3)

cbQosPoliceCfgTable
cbQosPoliceCfgEntry.*cbQosConfigIndex*

cbQosPoliceCfgEntry.1046

cbQosPoliceCfgRate	1000000
cbQosPoliceCfgBurstSize	8000
cbQosPoliceCfgExtBurstSize	8000
cbQosPoliceCfgConformAction	transmit(1)
cbQosPoliceCfgConformSetValue	0
cbQosPoliceCfgExceedAction	drop(5)
cbQosPoliceCfgExceedSetValue	0
cbQosPoliceCfgViolateAction	0
cbQosPoliceCfgViolateSetValue	0

69738

この QoS 設定例について、次の点に注意してください。

- ポリシー マップ pm-1Meg は、入力インターフェイスと出力インターフェイスに付加されているので、
  - 2 つの cbQosObjectsIndex 値が使用されます（入力インターフェイスおよび出力インターフェイスにそれぞれ 1 つずつ）。
  - 入力オブジェクトおよび出力オブジェクトの両方に 1 つの cbQosConfigIndex を使用します。
- 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。したがって、cm1 は cbQosCMCfgTable に含まれていません。

図 A-11 に、MIB テーブルに保存された RED 設定情報の例を示します。この設定は、GigabitEthernet ではなく、ATM インターフェイスに適用されます。

図 A-11　ATM QoS — RED コンフィギュレーション オブジェクト

QoS の監視

ここでは、表 A-3 に示す MIB テーブルの QoS 統計情報をチェックすることによって、Cisco 10000 シリーズ ESR 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-44）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-3　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 ステートメント — 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、機能のステート、およびキュー サイズ。
cbQosREDClassStatsTable	RED — 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、および送信されたバイト数およびオクテット数。

QoS 統計の処理に関する考慮事項

Cisco 10000 シリーズ ESR 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。以下の図では、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 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 統計情報テーブル

ここでは、CISCO-CLASS-BASED-QOS-MIB 統計情報テーブルのカウンタを一連の図で示します。

- 図 A-12 は、cbQosCMStatsTable のカウンタと、これらの統計情報およびその他の統計情報にアクセスするためのインデックスを示しています。
- 図 A-13 は、cbQosMatchStmtStatsTable、cbQosPoliceStatsTable、cbQosQueueingStatsTable、cbQosTSStatsTable、および cbQosREDClassStatsTable のカウンタを示しています。

テーブルに保存された QoS 統計情報の例は、「QoS 統計情報の例」（p.A-37）を参照してください。

以下の図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。

図 A-12　QoS クラス マップの統計情報およびインデックス

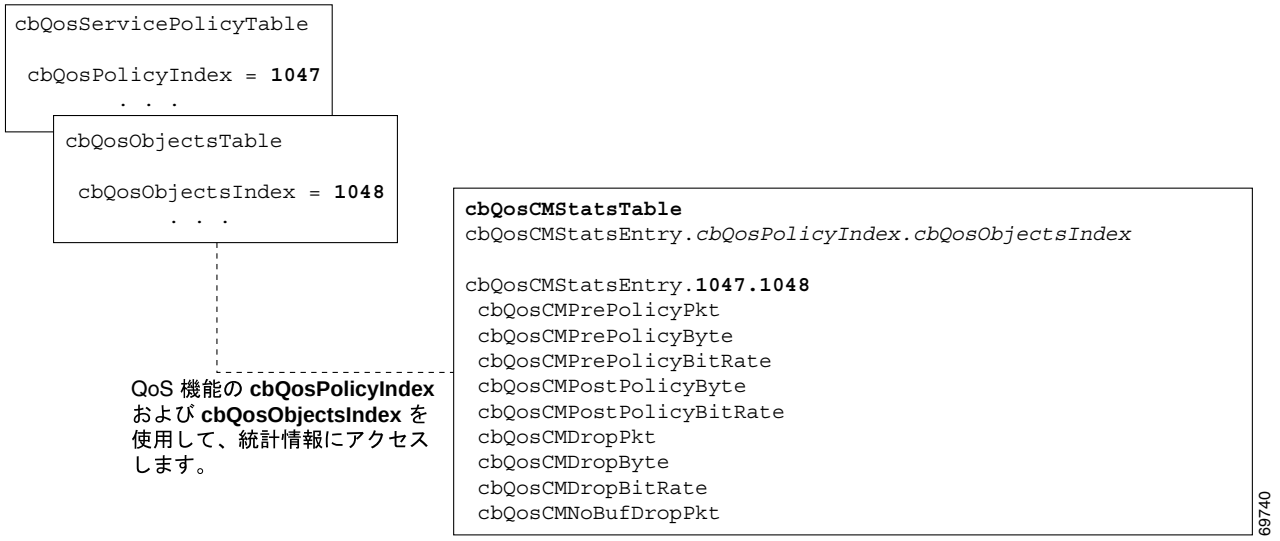
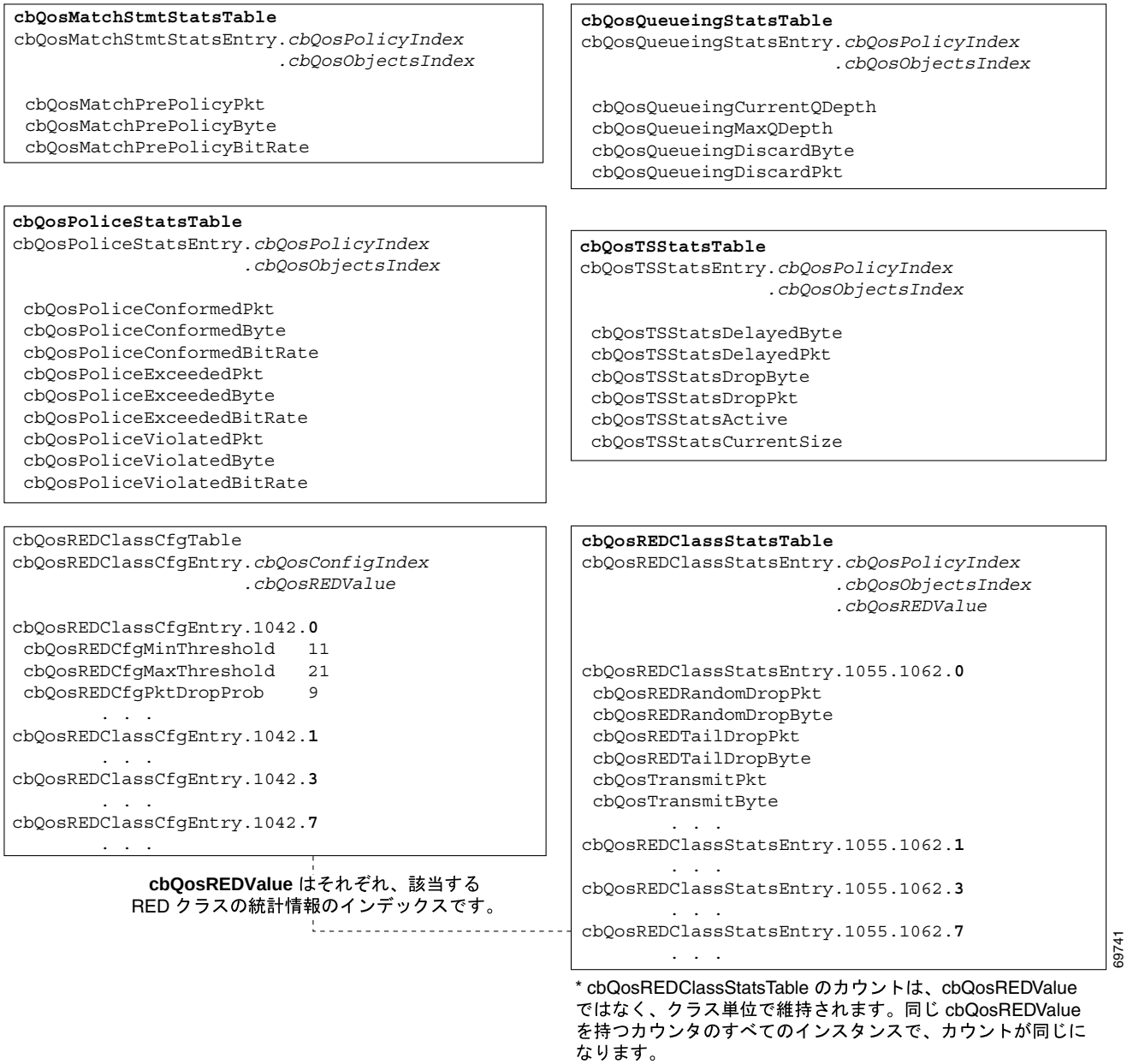


図 A-13　QoS 統計情報テーブル



* cbQosREDClassStatsTable のカウントは、cbQosREDValue ではなく、クラス単位で維持されます。同じ cbQosREDValue を持つカウンタのすべてのインスタンスで、カウントが同じになります。

QoS 統計情報の例

ここでは、show コマンドで表示される QoS 統計情報と、CISCO-CLASS-BASED-QOS-MIB テーブルに保存された QoS 統計情報の例を一連の図で示します。

- 図 A-14 は、**show policy-map interface** コマンドによる QoS 統計情報の出力例を示しています。
- 図 A-15 は、入力サービス ポリシーに対応するクラス マップ統計情報を示しています。
- 図 A-16 は、出力サービス ポリシーに対応するクラス マップ統計情報を示しています。
- 図 A-17 は、入力および出力サービス ポリシーに対応する match ステートメント統計情報を示しています。

図 A-14　QoS 統計情報の出力例 — CLI の show コマンド

```
c10k# show running-config interface GigabitEthernet 1/0/0.1
Building configuration...

Current configuration : 188 bytes
!
interface GigabitEthernet1/0/0.1
 encapsulation dot1Q 1
 ip address 10.1.0.2 255.255.255.0
 no ip directed-broadcast
 service-policy input pm-1meg
 service-policy output pm-1meg
end

c10k# show policy-map interface
 GigabitEthernet1/0/0.1

  Service-policy input: pm-1meg (2428)

    Class-map: class-default (match-any) (2429/0)
      4801508 packets, 667409423 bytes
      5 minute offered rate 1668000 bps, drop rate 667000 bps
      Match: any  (2431)
        4801508 packets, 667409423 bytes
        5 minute rate 1668000 bps
      Police:
        1000000 bps, 8000 limit, 8000 extended limit
        conformed 2878916 packets, 400169135 bytes; action: transmit
        exceeded 1922592 packets, 267240288 bytes; action: drop

  Service-policy output: pm-1meg (2433)

    Class-map: class-default (match-any) (2434/0)
      14259374 packets, 1925015267 bytes
      5 minute offered rate 1639000 bps, drop rate 640000 bps
      Match: any  (2436)
        14259374 packets, 1925015267 bytes
        5 minute rate 1639000 bps
      Output queue: 0/8192; 3698585/514006021 packets/bytes output, 0 drops
      Police:
        1000000 bps, 8000 limit, 8000 extended limit
        conformed 3517209 packets, 474822992 bytes; action: transmit
         exceeded 10742165 packets, 1450192275 bytes; action: drop c10k#
```

69742

図 A-15　QoS クラス マップ統計情報 — 入力サービス ポリシー

```
c10k# show policy-map interface
 GigabitEthernet1/0/0.1

  Service-policy input: pm-1meg (2428)

    Class-map: class-default (match-any) (2429/0)
      4801508 packets, 667409423 bytes
      5 minute offered rate 1668000 bps, drop rate 667000 bps
      Match: any  (2431)
        4801508 packets, 667409423 bytes
        5 minute rate 1668000 bps
      Police:
        1000000 bps, 8000 limit, 8000 extended limit
        conformed 2878916 packets, 400169135 bytes; action: transmit
        exceeded 1922592 packets, 267240288 bytes; action: drop
```

```
cbQosCMCfgTable
 cbQosCMName = class-default
 cbQosCMInfo = matchAny(3)

cbQosObjectsTable
 cbQosObjectsIndex = 2004
 cbQosObjectsType = classmap(2)
 cbQosParentObjectsIndex = 2003

cbQosServicePolicyTable
 cbQosPolicyIndex = 2003
 cbQosIfType = subinterface(2)
 cbQosPolicyDirection = input(1)
```

```
cbQosCMStatsTable
cbQosCMStatsEntry.cbQosPolicyIndex.cbQosObjectsIndex

cbQosCMStatsEntry.2003.2004
 cbQosCMPrePolicyPktOverflow      0
 cbQosCMPrePolicyPkt              4801508
 cbQosCMPrePolicyPkt64            0x0004943e4
 cbQosCMPrePolicyByteOverflow     0
 cbQosCMPrePolicyByte             667409423
 cbQosCMPrePolicyByte64           0x027c7dc0f
 cbQosCMPrePolicyBitRate          1668000
 cbQosCMPostPolicyByteOverflow    0
 cbQosCMPostPolicyByte            401004108
 cbQosCMPostPolicyByte64          0x017e6d64c
 cbQosCMPostPolicyBitRate         1001000
 cbQosCMDropPktOverflow           0
 cbQosCMDropPkt                   1922592
 cbQosCMDropPkt64                 0x0001d5620
 cbQosCMDropByteOverflow          0
 cbQosCMDropByte                  266405315
 cbQosCMDropByte64                0x00fe105c3
 cbQosCMDropBitRate               667000
 cbQosCMNoBufDropPktOverflow      0
 cbQosCMNoBufDropPkt              0
 cbQosCMNoBufDropPkt64            0x000000000
```

69743

図 A-16　QoS クラス マップ統計情報 — 出力サービス ポリシー

図 A-17　QoS match ステートメント統計情報

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際、これらの例を参考にしてください。このコード例では、次の処理を行います。

- 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 課金情報の検索

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 サービスの課金業務などの) アプリケーション処理を行えるよう、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。Cisco 10000 シリーズ ESR 上に 2000 個のカスタマー インターフェイスがある場合、各インターフェイスに対応づ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別するには、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ため、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 ポリシーが対応づ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カスタマー インターフェイスをマークします。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取り扱いを指定します。

```
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 課金情報の検索

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 を検索
- 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 インデックスを検索
- 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検索。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。

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
```

課金情報を検索します。

```
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調べます。

```
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。

ここでは次の内容を説明します。

出入力インターフェイスのカウント

Cisco 10000 シリーズ ESR は、出入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、ESR はインターフェイス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

以下のステップを実行し、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します。

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　（任意）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説明します。

シナリオを作成するためには、次の手順を実行します。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発生させます。このトラフィックにはサービス ポリシーが適用されています。
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できません。

（注） このシナリオでは、Cisco 10000 シリーズ ESR 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 10000 Series ESR 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 カウントを確認

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です。

CLI コマンド出力

```
c10k# 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

```
c10k# 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、今回のシナリオのトラフィックが生成されます。

```
c10k# 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 カウントを確認

ping コマンドを使ってトラフィックを生成したら、**police** コマンドで設定した Committed Access Rate（CAR; 専用アクセス レート）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です（準拠した [conformed] パケットと超過した [exceeded] パケットの数は太字で示されています）。

CLI コマンド出力

```
c10k# show policy-map interface g6/0/0.10

GigabitEthernet6/0/0.10

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

```
c10k# getmany -v2c 10.86.0.63 public ciscoCBQosMIB
        . . .
      cbQosCMDropPkt.1143.1145 = 57
        . . .
      cbQosPoliceConformedPkt.1143.1151 = 42
        . . .
```