

Exemplo de configuração de MIB de expressão e MIB de evento

Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Componentes Utilizados](#)

[Conventions](#)

[Informações de Apoio](#)

[Configurar](#)

[MIB de expressão](#)

[MIB de evento](#)

[Verificar](#)

[Troubleshoot](#)

[Comandos para Troubleshooting](#)

[Informações Relacionadas](#)

[Introduction](#)

Este documento mostra como combinar o MIB de expressão e o MIB de evento para uso no gerenciamento de falhas. O exemplo incluído não é realista, mas mostra muitos recursos disponíveis.

O roteador deve executar duas ações:

1. Envie uma armadilha se uma interface de loopback tiver uma largura de banda superior a 100 e estiver administrativamente inativa
2. A interface de loopback será desligada se uma das interfaces tiver sua instrução de largura de banda alterada de um valor definido

O exemplo é mostrado com o status de largura de banda e de admin porque eles são fáceis de manipular a partir da linha de comando e ambos mostram valores inteiros e booleanos.

Os comandos neste documento usam o parâmetro Object Identifier (OID) e não os nomes dos objetos. Isso permite o teste sem carregar a MIB.

[Prerequisites](#)

[Requirements](#)

Antes de usar as informações neste documento, certifique-se de que você atenda aos seguintes

pré-requisitos:

- A estação de trabalho deve ter ferramentas SNMP (Simple Network Management Protocol) fornecidas pela Hewlett-Packard (HP) Openview. Outras ferramentas SNMP funcionam, mas podem ter sintaxe diferente.
- O dispositivo deve executar o software Cisco IOS® versão 12.2(4)T3 ou posterior. As versões anteriores não suportam a versão RFC do MIB de evento.
- A plataforma deve suportar o MIB de evento. Para obter uma lista de plataformas suportadas para o Cisco IOS Software Release 12.1(3)T, consulte a seção "Plataforma Suportada" do [Suporte MIB de Evento](#).

Componentes Utilizados

As informações neste documento são baseadas nestas versões de software e hardware:

- Software Cisco IOS versão 12.3(1a)
- Roteador de acesso modular Cisco 3640

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the [Cisco Technical Tips Conventions](#).

Informações de Apoio

- A MIB de expressão permite que o usuário crie seu próprio objeto MIB com base em uma combinação de outros objetos. Para obter mais informações, consulte o [RFC 2982](#).
- A MIB de evento permite que o usuário tenha o dispositivo monitorando seus próprios objetos MIB e gere ações (comandos de notificação ou **SNMP SET**) com base em um evento definido. Para obter mais informações, consulte o [RFC 2981](#).

Configurar

Observação: algumas das linhas do código de saída são exibidas em duas linhas para se ajustarem melhor à sua tela.

Neste exemplo, ifIndex da interface de loopback é igual a 16.

```
# snmpget -v 2c -c private router .1.3.6.1.2.1.2.2.1.2.16
IF-MIB::ifDescr.16 = STRING: Loopback0
```

Os nomes das variáveis relacionadas ao primeiro evento começam com e1 e os relacionados ao segundo início com e2. O nome do roteador é "router" e a string de comunidade de leitura/gravação é "private".

MIB de expressão

Criando a expressão 1

Primeiro, crie uma expressão que retorne um valor de 1 se a condição, `ifSpeed` for maior que 100.000 AND `seAdminStatus` estiver inativo para a interface de loopback. Se a condição não for atendida, ela retornará o valor 0.

1. [expExpressionDeltaInterval](#) — Este objeto não é usado. Não há motivo para calcular uma expressão quando ela não é interrogada. Se nenhum valor for definido, a expressão será calculada quando o objeto for consultado. O nome da expressão é `e1exp`, que na tabela ASCII corresponde a 101 49 101 120 112.

2. [expNameStatus](#) — Isso destrói uma eventual expressão antiga criada.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.49.101.120.112 integer 6
```

3. [expNameStatus](#) — Criar e aguardar.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.49.101.120.112 integer 5
```

4. [expExpressionIndex](#) — Isto cria o índice a ser usado posteriormente para recuperar o resultado da expressão.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.2.101.49.101.120.112 gauge 1
```

5. [expExpressionComment](#)—Aqui .1 (o `expExpressionIndex` escolhido) é a descrição da expressão.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.1.1.4.1 octetstring "e1 expression"
```

6. [expExpression](#) —Esta é a expressão em si, as variáveis \$1 e \$2 são definidas na próxima etapa. Os únicos operadores permitidos são (para obter detalhes, consulte [RFC 2982](#)):

() - (unary) + - * / % & | ^ << >> ~ ! && || == != > >= < <=

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.1.1.2.1 octetstring '$1 < 100000 && $2 == 2'
```

7. [expObjectID](#)

```
.1 is for the variable $1 => ifSpeed  
.2 for $2 => ifAdminStatus
```

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.2.1.1 objectidentifier 1.3.6.1.2.1.2.2.1.5.16
```

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.2.1.2 objectidentifier 1.3.6.1.2.1.2.2.1.7.16
```

8. [expObjectSampleType](#) — Os dois valores são tomados em valores absolutos (para Delta, considere 2 como o valor).

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.4.1.1 integer 1
```

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.4.1.2 integer 1
```

9. [expObjectIDWildcard](#) — As IDs de objeto não são curinga. Esse é o valor padrão, portanto, não `snmpset expObjectIDWildcard`.

10. [expObjectStatus](#) — Defina as linhas em `expObjectTable` como ativo.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.10.1.1 integer 1
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.10.1.2 integer 1
```

11. Ative a expressão 1.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.49.101.120.112 integer 1
```

Testando a expressão 1

```
router(config)#interface loopback 0
router(config-if)#shutdown
router(config-if)#bandwidth 150
```

1. Se a condição for atendida, o valor de [expValueCounter32Val](#) é 1 (como o valor de [expExpressionValueType](#) permanece inalterado, o resultado é um contador32). **Observação:** o tipo não pode ser um valor de ponto flutuante.

```
# snmpwalk -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.4.1.1.2
cisco.ciscoExperiment.22.1.4.1.1.2.1.0.0.0 : Counter: 1
```

```
router(config-if)#bandwidth 150000
```

2. Se a condição não for atendida, o valor será 0.

```
# snmpwalk -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.4.1.1.2
cisco.ciscoExperiment.22.1.4.1.1.2.1.0.0.0 : Counter: 0
```

```
router(config-if)#bandwidth 1
router(config-if)#no shutdown
```

3. Se a condição não for atendida, o valor será 0.

```
# snmpwalk -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.4.1.1.2
cisco.ciscoExperiment.22.1.4.1.1.2.1.0.0.0 : Counter: 0
```

Criando e Testando a Expressão 2

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.50.101.120.112 integer 6
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.50.101.120.112 integer 5
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.2.101.50.101.120.112 gauge 2
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.1.1.4.2 octetstring "e2 expression"
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.1.1.2.2 octetstring '($1 * 18) / 23'
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.2.2.1 objectidentifier
1.3.6.1.2.1.2.2.1.5
```

1. [expObjectIDWildcard](#) — Indica que `1.3.6.1.2.1.2.2.1.5` é uma tabela e não um objeto.

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.3.2.1 integer 1
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.4.2.1 integer 1
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.3.2.1.10.2.1 integer 1
```

```
# snmpset -v 2c -c private router 1.3.6.1.4.1.9.10.22.1.2.3.1.3.101.50.101.120.112 integer 1
```

2. Teste:

```
# snmpwalk router 1.3.6.1.4.1.9.10.22.1.4.1.1
[...]
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.10 : Counter: 0
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.11 : Counter: 23250000
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.12 : Counter: 42949672
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.13 : Counter: 18450
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.14 : Counter: 150
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.15 : Counter: 1350
cisco.ciscoExperiment.22.1.4.1.1.2.2.0.0.16 : Counter: 9600
```

MIB de evento

Criando evento 1

Agora, crie um evento que verifique o valor de saída da primeira expressão a cada 60 segundos e o compara com uma referência. Quando a referência corresponde ao valor da expressão, uma armadilha é disparada com o VARBIND escolhido.

1. Crie o gatilho na tabela de gatilho. O nome do gatilho é trigger1, que no código ASCII é 116 114 105 103 103 101 114 49. O proprietário é tom: 116 111 109. O índice de mteTriggerEntry é composto pelo proprietário do gatilho e pelo nome do gatilho. O primeiro valor do índice fornece o número de caracteres para mteOwner. Nesse caso, há três caracteres para tom, portanto, o índice é: 3.116.111.109.116.114.105.103.103.101.114.49.
2. Destrua a entrada antiga se ela existir.
3. Defina o status do acionador para **criar e aguardar**.
4. A última etapa o ativa: [mteTriggerEntryStatus](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.49
integer 6
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.49
integer 5
```

[mteTriggerValueID](#) — O valor da primeira expressão é e_{1exp} . O identificador de objeto do objeto MIB é o que deve ser amostrado para ver se o disparador deve disparar.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.6.3.116.111.109.116.114.105.103.103.101.114.49
objectIdentifier
1.3.6.1.4.1.9.10.22.1.4.1.1.2.1.0.0.0
```

[mteTriggerValueIDWildcard](#) — sem usar um curinga para a ID do valor.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.7.3.116.111.109.116.114.105.103.103.101.114.49
integer 2
```

[mteTriggerTest](#) — Existência (0), booleano (1) e limiar (2). O método para selecionar um dos valores acima é complexo. Para selecionar uma existência, forneça um valor em oito dígitos em que o primeiro é 1, como 10000000 ou 100xxxxx. Para um booleano, o segundo dígito deve ser um 1: 01000000 ou 010xxxxx. Para um limite, o terceiro dígito deve ser 1: 00100000 ou

001xxxxx. É mais fácil trabalhar assim: Para a existência, o valor é octetstringhex—80. Para booleano, o valor é octetstringhex—40. Para o limite, o valor é octetstringhex—20.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.4.3.116.111.109.116.114.105.103.103.101.114.49
octetstringhex "40"
```

[mteTriggerFrequency](#) — Determina o número de segundos a aguardar entre as amostras de disparo. O valor mínimo é definido com o objeto mteResourceSampleMinimum (o padrão é 60 segundos), a redução desse valor aumenta o uso da CPU, portanto, deve ser feita com cuidado.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.11.3.116.111.109.116.114.105.103.103.101.114.49
gauge 60
```

[mteTriggerSampleType](#) — Estes são absoluteValue (1) e deltaValue (2). Nesse caso, o valor é absoluto:

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.5.3.116.111.109.116.114.105.103.103.101.114.49
integer 1
```

[mteTriggerEnabled](#) — Este é um controle que permite que um gatilho seja configurado, mas não usado. Defina como verdadeiro (o padrão é falso).

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.14.3.116.111.109.116.114.105.103.103.101.114.49
integer 1
```

Agora que o gatilho foi criado, defina o evento que o gatilho usará. O nome do evento é event1.[mteEventEntryStatus](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.49
integer 6
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.49
integer 5
```

[mteEventActions](#) — são notificação (0) e definido (1). O processo é o mesmo para mteTriggerTest. A notificação é 10xxxxxxx e definida como 01xxxxxxx.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.3.3.116.111.109.101.118.101.110.116.49
octetstringhex "80"
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.4.3.116.111.109.101.118.101.110.116.49
integer 1
```

Esta próxima etapa define o teste a ser feito no objeto selecionado para trigger1.[mteTriggerBooleanComparison](#) — São diferentes (1), iguais (2), menores (3), menores (4), maiores (5) e maiores (6). Nesse caso, igual.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.1.3.116.111.109.116.114.105.103.103.101.114.49
integer 2
```

[mteTriggerBooleanValue](#) — Este é o valor a ser usado para o teste. Se o valor de 1.3.6.1.4.1.9.10.22.1.4.1.1.2.1.0.0.0 for igual a 1, a condição é atendida.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.2.3.116.111.109.116.114.105.103.103.101.114.49
```

```
integer 1
```

Agora defina o objeto a ser enviado com o evento. [mteTriggerBooleanObjectsOwner](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.4.3.116.111.109.116.114.105.103.103.101.114.49
octetstring "tom"
```

[mteTriggerBooleanObjects](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.5.3.116.111.109.116.114.105.103.103.101.114.49
octetstring "objects1"
```

[mteTriggerBooleanEventOwner](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.6.3.116.111.109.116.114.105.103.103.101.114.49
octetstring "tom"
```

[mteTriggerBooleanEvent](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.5.1.7.3.116.111.109.116.114.105.103.103.101.114.49
octetstring "event1"
```

Crie a tabela de objetos. Envie o valor de 1.3.6.1.2.1.2.2.1.5.16 como VARBIND com a armadilha. Object Table [mteObjectsName](#) — Objects1. [mteObjectsEntryStatus](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.3.1.1.5.3.116.111.109.8.111.98.106.101.99.116.115.49.1
integer 6
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.3.1.1.5.3.116.111.109.8.111.98.106.101.99.116.115.49.1
integer 5
```

[mteObjectsID](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.3.1.1.3.3.116.111.109.8.111.98.106.101.99.116.115.49.1
objectidentifier 1.3.6.1.2.1.2.2.1.5.16
```

[mteObjectsIDWildcard](#) — Não há curinga usada.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.3.1.1.4.3.116.111.109.8.111.98.106.101.99.116.115.49.1
integer 1
```

Ative a tabela de objetos.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.3.1.1.5.3.116.111.109.8.111.98.106.101.99.116.115.49.1
integer 1
```

Anexe o objeto ao evento1. [Notifique mteEventName](#) —
Evento1. [mteEventNotificationObjectsOwner](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.3.1.2.3.116.111.109.101.118.101.110.116.49
octetstring "tom"
```

[mteEventNotificationObjects](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.3.1.3.3.116.111.109.101.118.101.110.116.49
octetstring "objects1"
```

Ative o gatilho.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.49
integer 1
```

Ativar o evento.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.49
integer 1
```

Armadilha recebida

```
Enterprise : 1.3.6.1.2.1.88.2
Trap type : ENTERPRISE SPECIFIC (6)
Specific trap type: 1
object 1 : mteHotTrigger
value : STRING: "trigger1"
object 2 : mteHotTargetName
value: ""
object 3 : mteHotContextName
value: ""
object 4: mteHotOID
value: OID: 1.3.6.1.4.1.9.10.22.1.4.1.1.2.1.0.0.0
object 5: mteHotValue
value: INTEGER: 1
object 6: 1.3.6.1.2.1.2.2.1.5.16
value: Gauge32: 1000
```

Observação: o objeto 6 é a VARBIND que foi adicionada.

Criando evento 2

Siga estes passos:

1. mteTriggerName — Trigger2.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.50
integer 6
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.50
integer 5
```

2. mteTriggerValueID — Este é o valor da primeira expressão e mteTriggerValueIDWildcard. Dessa vez, o processo corta o ID do valor, o identificador de objeto do objeto MIB a ser amostrado para determinar se o disparador é disparado.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.6.3.116.111.109.116.114.105.103.103.101.114.50
objectidentifier
1.3.6.1.4.1.9.10.22.1.4.1.1.2.2.0.0
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.7.3.116.111.109.116.114.105.103.103.101.114.50
integer 1
```

3. mteTriggerTest — Limite.


```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.4.3.116.111.109.116.114.105.103.103.101.114.50
octetstringhex "20"
```

4. [mteTriggerFrequency](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.11.3.116.111.109.116.114.105.103.103.101.114.50
gauge 60
```

5. [mteTriggerSampleType](#) — Valor Delta.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.5.3.116.111.109.116.114.105.103.103.101.114.50
integer 2
```

6. [mteTriggerEnabled](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.14.3.116.111.109.116.114.105.103.103.101.114.50
integer 1
```

7. Crie um evento na tabela de eventos // [mteEventName](#)—evento2.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.50
integer 6
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.50
integer 5
```

8. [mteEventActions](#) — O valor 40 é para Set, o que significa que quando a condição é atendida, o roteador emite um comando **snmp set**. Nesse caso, ele faz o Set para si mesmo, mas também pode fazer a operação em um dispositivo remoto.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.3.3.116.111.109.101.118.101.110.116.50
octetstringhex "40"
```

9. Ative o evento.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.4.3.116.111.109.101.118.101.110.116.50
integer 1
```

10. Defina o Limite de disparo na Tabela de disparo // índice =

[mteTriggerName](#)—Trigger2.Como se trata de um limiar, indicar valores para condições de falha e de aumento. Vejam só a situação de elevação desta vez.

11. [mteTriggerThresholdDeltaRising](#) — Este é o valor limite a ser verificado.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.6.1.4.3.116.111.109.116.114.105.103.103.101.114.50
integer 100
```

12. [mteTriggerThresholdDeltaRisingEventOwner](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.6.1.12.3.116.111.109.116.114.105.103.103.101.114.50
octetstring "tom"
```

13. [mteTriggerThresholdDeltaRisingEvent](#)

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.6.1.13.3.116.111.109.116.114.105.103.103.101.114.50
octetstring "event2"
```

14. [mteEventSetObject](#) — Este é o identificador de objeto do objeto MIB a ser definido. Aqui, ifAdminStatus para a interface de loopback.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.4.1.1.3.116.111.109.101.118.101.110.116.50
objectidentifier 1.3.6.1.2.1.2.2.1.7.16
```

15. [mteEventSetValue](#) — Este é o valor a ser definido (2 para baixo).

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.4.1.3.3.116.111.109.101.118.101.110.116.50
integer 2
```

16. Ative o gatilho.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.2.2.1.15.3.116.111.109.116.114.105.103.103.101.114.50
integer 1
```

17. Ativar o evento.

```
# snmpset -v 2c -c private router
1.3.6.1.2.1.88.1.4.2.1.5.3.116.111.109.101.118.101.110.116.50
integer 1
```

[Resultado](#)

```
router(config)#int lo1
router(config-if)#bandwidth 5000000
16:24:11: %SYS-5-CONFIG_I: Configured from 10.48.71.71 by snmp
16:24:13: %LINK-5-CHANGED: Interface Loopback1, changed state to administratively down
16:24:14: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1, changed state to down
```

Observação: aqui, 10.48.71.71 é o endereço do próprio roteador.

[Verificar](#)

Esta seção fornece informações a serem usadas para confirmar se a configuração está funcionando corretamente.

A [Output Interpreter Tool \(somente clientes registrados\)](#) oferece suporte a determinados comandos show, o que permite exibir uma análise da saída do comando show.

```
router #show management event
Mgmt Triggers:
(1): Owner: tom
(1): trigger1, Comment: , Sample: Abs, Freq: 15
    Test: Boolean
    ObjectOwner: , Object:
    OID: ciscoExperiment.22.1.4.1.1.2.1.0.0.0, Enabled 1, Row Status 1
    Boolean Entry:
    Value: 1, Cmp: 2, Start: 1
    ObjOwn: tom, Obj: objects1, EveOwn: tom, Eve: event1

Delta Value Table:
(0): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.1.0.0.0 , val: 0
```

(2): trigger2, Comment: , Sample: Del, Freq: 60
Test: Threshold
ObjectOwner: , Object:
OID: ciscoExperiment.22.1.4.1.1.2.2.0.0, Enabled 1, Row Status 1
Threshold Entry:
Rising: 0, Falling: 0, DeltaRising: 100, DeltaFalling: 0
ObjOwn: , Obj:
RisEveOwn: , RisEve: , FallEveOwn: , FallEve:
DelRisEveOwn: tom, DelRisEve: event2, DelFallEveOwn: , DelFallEve:

Delta Value Table:

(0): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.1 , val: 62000000
(1): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.2 , val: 4000000
(2): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.3 , val: 617600
(3): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.4 , val: 617600
(4): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.5 , val: 617600
(5): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.6 , val: 617600
(6): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.7 , val: 858993458
(7): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.8 , val: 0
(8): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.9 , val: 62000000
(9): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.10 , val: 0
(10): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.11 , val: 62000000
(11): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.12 , val: 858993458
(12): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.13 , val: 858993458
(13): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.14 , val: 400
(14): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.15 , val: 3600
(15): Thresh: , Exis: 1, Read: 0, OID: ciscoExperiment.22.1.4.1.1.2.2.0.0.16 , val: 25600

Mgmt Events:

(1): Owner: tom
(1)Name: event1, Comment: , Action: Notify, Enabled: 1 Status: 1
Notification Entry:
ObjOwn: tom, Obj: objects1, OID: ccitt.0
(2)Name: event2, Comment: , Action: Set, Enabled: 1 Status: 1
Set:
OID: ifEntry.7.13, SetValue: 2, Wildcard: 2
TAG: , ContextName:

Object Table:

(1): Owner: tom
(1)Name: objects1, Index: 1, OID: ifEntry.5.13, Wild: 2, Status: 1

Failures: Event = 44716, Trigger = 0

router #show management expression

Expression: e1exp is active
Expression to be evaluated is \$1 < 100000 && \$2 == 2 where:
\$1 = ifEntry.5.13
Object Condition is not set
Sample Type is absolute
Both ObjectID and ObjectConditional are not wildcarded
\$2 = ifEntry.7.13
Object Condition is not set
Sample Type is absolute
Both ObjectID and ObjectConditional are not wildcarded

Expression: e2exp is active
Expression to be evaluated is (\$1 * 18) / 23 where:
\$1 = ifEntry.5
Object Condition is not set
Sample Type is absolute
ObjectID is wildcarded

Troubleshoot

Esta seção fornece informações a serem usadas para solucionar problemas de configuração.

Comandos para Troubleshooting

Estes são os comandos para ativar a depuração:

```
router#debug management expression mib  
router#debug management event mib
```

Observação: antes de emitir comandos **debug**, consulte [Informações Importantes sobre Comandos Debug](#).

Informações Relacionadas

- [MIB de expressão: RFC 2982](#)
- [MIB de evento: RFC 2981](#)
- [EXPRESSION-MIB.my / EVENT-MIB.my](#)
- [Guia de recursos do IOS: Suporte a MIB de evento](#)
- [Suporte Técnico - Cisco Systems](#)