

Comment ajouter, modifier et supprimer les VLAN sur Catalyst à l'aide de SNMP

Contenu

[Introduction](#)

[Conditions préalables](#)

[Conditions requises](#)

[Composants](#)

[Conventions](#)

[Fond](#)

[Détails des variables MIB, y compris les identifiants d'objet \(OID\)](#)

[Ajouter un réseau VLAN à un commutateur Cisco Catalyst avec SNMP](#)

[Step-by-Step Instructions](#)

[Ajouter un réseau VLAN à un commutateur Cisco Catalyst avec SNMP](#)

[Instructions en une étape](#)

[Supprimer un réseau VLAN d'un commutateur Cisco Catalyst avec SNMP](#)

[Step-by-Step Instructions](#)

[Ajouter un port à un réseau VLAN sur un commutateur Cisco Catalyst avec SNMP](#)

[Comment changer un port d'un réseau VLAN à un autre](#)

[Informations connexes](#)

[Introduction](#)

Ce document explique comment créer et supprimer des VLAN sur un commutateur Cisco Catalyst qui utilise le protocole SNMP. Il décrit également comment ajouter des ports à un VLAN avec le protocole SNMP.

[Conditions préalables](#)


[Conditions requises](#)

Avant d'utiliser le contenu de ce document, assurez-vous de maîtriser ce qui suit :

- Le fonctionnement de l'ifTable et de l'ifIndex
- Le fonctionnement des réseaux VLAN sur les commutateurs Cisco Catalyst
- L'affichage des renseignements du réseau VLAN sur les commutateurs Cisco Catalyst
- L'utilisation générale des commandes SNMP **get**, **set** et **walk**

[Composants](#)

Ce document traite des commutateurs Catalyst qui exécutent des systèmes d'exploitation Catalyst OS ou Catalyst IOS compatibles avec IF-MIB, CISCO-VTP-MIB et CISCO-VLAN-MEMBERSHIP-MIB. Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Catalyst 3524XL exécutant CatIOS 12.0(5)WC5a
- NET-SNMP version 5.0.6 disponible au <http://www.net-snmp.org/> 

Les informations présentées dans ce document ont été créées à partir de périphériques dans un environnement de laboratoire spécifique. All of the devices used in this document started with a cleared (default) configuration. Si vous travaillez sur un réseau qui fonctionne en permanence, assurez-vous de bien comprendre l'incidence possible d'une commande avant de l'utiliser.

Conventions

Pour plus d'informations sur les conventions des documents, référez-vous aux [Conventions utilisées pour les conseils techniques de Cisco](#).

Fond

Détails des variables MIB, y compris les identifiants d'objet (OID)

1.3.6.1.4.1.9.9.46.1.3.1.1.2 (CISCO-VTP-MIB)

```
vtpVlanState OBJECT-TYPE
    SYNTAX      INTEGER { operational(1),
                        suspended(2),
                        mtuTooBigForDevice(3),
                        mtuTooBigForTrunk(4) }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION  "The state of this VLAN.
```

The state 'mtuTooBigForDevice' indicates that this device cannot participate in this VLAN because the VLAN's MTU is larger than the device can support.

The state 'mtuTooBigForTrunk' indicates that while this VLAN's MTU is supported by this device, it is too large for one or more of the device's trunk ports."

```
::= { vtpVlanEntry 2 }
```

1.3.6.1.4.1.9.9.46.1.4.1.1.1 (CISCO-VTP-MIB)

```
vtpVlanEditOperation OBJECT-TYPE
    SYNTAX      INTEGER { none(1),
                        copy(2),
                        apply(3),
                        release(4),
                        restartTimer(5)
                    }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION  "This object always has the value 'none' when read. When
                written, each value causes the appropriate action:
```

'copy' - causes the creation of rows in the

vtpVlanEditTable exactly corresponding to the current global VLAN information for this management domain. If the Edit Buffer (for this management domain) is not currently empty, a copy operation fails. A successful copy operation starts the deadman-timer.

'apply' - first performs a consistent check on the the modified information contained in the Edit Buffer, and if consistent, then tries to instanciate the modified information as the new global VLAN information. Note that an empty Edit Buffer (for the management domain) would always result in an inconsistency since the default VLANs are required to be present.

'release' - flushes the Edit Buffer (for this management domain), clears the Owner information, and aborts the deadman-timer. A release is generated automatically if the deadman-timer ever expires.

'restartTimer' - restarts the deadman-timer.

'none' - no operation is performed."

::= { vtpEditControlEntry 1 }

1.3.6.1.4.1.9.9.46.1.4.1.1.3 (CISCO-VTP-MIB)

vtpVlanEditBufferOwner OBJECT-TYPE

SYNTAX OwnerString

MAX-ACCESS read-create

STATUS current

DESCRIPTION "The management station which is currently using the Edit Buffer for this management domain. When the Edit Buffer for a management domain is not currently in use, the value of this object is the zero-length string. Note that it is also the zero-length string if a manager fails to set this object when invoking a copy operation."

::= { vtpEditControlEntry 3 }

1.3.6.1.4.1.9.9.46.1.4.2.1.11 (CISCO-VTP-MIB)

vtpVlanEditRowStatus OBJECT-TYPE

SYNTAX RowStatus

1:active

2:notInService

3:notReady

4:createAndGo

5:createAndWait

6:destroy

MAX-ACCESS read-create

STATUS current

DESCRIPTION "The status of this row. Any and all columnar objects in an existing row can be modified irrespective of the status of the row.

A row is not qualified for activation until instances of at least its vtpVlanEditType, vtpVlanEditName and vtpVlanEditDot10Said columns have appropriate values.

The management station should endeavor to make all rows consistent in the table before 'apply'ing the buffer. An inconsistent entry in the table will cause the entire buffer to be rejected with the vtpVlanApplyStatus object set to the appropriate error value."

::= { vtpVlanEditEntry 11 }

1.3.6.1.4.1.9.9.46.1.4.2.1.3.1.48 (CISCO-VTP-MIB)

vtpVlanEditType OBJECT-TYPE
SYNTAX VlanType
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The type which this VLAN would have.
An implementation may restrict access to this object."
DEFVAL { ethernet }
::= { vtpVlanEditEntry 3 }

1.3.6.1.4.1.9.9.46.1.4.2.1.4.1.48 (CISCO-VTP-MIB)

vtpVlanEditName OBJECT-TYPE
SYNTAX DisplayString (SIZE (1..32))
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The name which this VLAN would have. This name would be
used as the ELAN-name for an ATM LAN-Emulation segment of
this VLAN.

An implementation may restrict access to this object."
::= { vtpVlanEditEntry 4 }

1.3.6.1.4.1.9.9.46.1.4.2.1.6.1.48 (CISCO-VTP-MIB)

vtpVlanEditDot10Said OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (4))
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The value of the 802.10 SAID field which would be used for
this VLAN.

An implementation may restrict access to this object."
::= { vtpVlanEditEntry 6 }

1.3.6.1.4.1.9.9.46.1.4.1.1.2.1 (CISCO-VTP-MIB)

vtpVlanApplyStatus OBJECT-TYPE
SYNTAX INTEGER { inProgress(1),
succeeded(2),
configNumberError(3),
inconsistentEdit(4),
tooBig(5),
localNVStoreFail(6),
remoteNVStoreFail(7),
editBufferEmpty(8),
someOtherError(9)
}
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The current status of an 'apply' operation to instantiate
the Edit Buffer as the new global VLAN information (for this
management domain). If no apply is currently active, the
status represented is that of the most recently completed
apply. The possible values are:

inProgress - 'apply' operation in progress;

succeeded - the 'apply' was successful (this value is
also used when no apply has been invoked since the
last time the local system restarted);

```

configNumberError - the apply failed because the value of
                    vtpVlanEditConfigRevNumber was less or equal to
                    the value of current value of
                    managementDomainConfigRevNumber;

inconsistentEdit - the apply failed because the modified
                  information was not self-consistent;

tooBig - the apply failed because the modified
         information was too large to fit in this VTP
         Server's non-volatile storage location;

localNVStoreFail - the apply failed in trying to store
                  the new information in a local non-volatile
                  storage location;

remoteNVStoreFail - the apply failed in trying to store
                  the new information in a remote non-volatile
                  storage location;

editBufferEmpty - the apply failed because the Edit
                 Buffer was empty (for this management domain).

someOtherError - the apply failed for some other reason
                (e.g., insufficient memory)."
 ::= { vtpEditControlEntry 2 }

```

1.3.6.1.4.1.9.9.68.1.2.2.1.2 (CISCO-VLAN-MEMBERSHIP-MIB)

```

vmVlan OBJECT-TYPE
    SYNTAX      INTEGER(0..4095)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION "The VLAN id of the VLAN the port is assigned to
                when vmVlanType is set to static or dynamic.
                This object is not instantiated if not applicable.

                The value may be 0 if the port is not assigned
                to a VLAN.

                If vmVlanType is static, the port is always
                assigned to a VLAN and the object may not be
                set to 0.

                If vmVlanType is dynamic the object's value is
                0 if the port is currently not assigned to a VLAN.
                In addition, the object may be set to 0 only."
 ::= { vmMembershipEntry 2 }

```

[Ajouter un réseau VLAN à un commutateur Cisco Catalyst avec SNMP](#)

[Step-by-Step Instructions](#)

Dans l'exemple ci-dessous, le réseau VLAN 11 est ajouté au commutateur :

1. Pour vérifier quels réseaux VLAN sont actuellement configurés sur le commutateur, émettez un **snmpwalk** sur l'OID de **vtpVlanState** : **Remarque** : Le dernier numéro de l'OID est le

numéro du réseau VLAN.

```
snmpwalk -c public crumpy vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1
.1 : INTEGER: operational
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1
.48 : INTEGER: operational
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1
.1002 : INTEGER: operational
```

2. Vérifiez si l'édition est utilisée par une autre station ou un autre appareil NMS. L'édition n'est pas utilisée si vous voyez ce message : Aucun objet MIB contenu dans la sous-arborescence :

```
snmpwalk -c public crumpy vtpVlanEditTable
no MIB objects contained under subtree.
```

3. L'édition n'est pas utilisée, il est donc prudent de commencer à la modifier. Réglez l'état de copie de **vtpVlanEditOperation** (nombre entier 2). Vous pourrez ainsi créer le réseau VLAN.

```
snmpset -c private crumpy vtpVlanEditOperation.1 integer 2
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControlEntry.vtpVlanEditOperation.1 : INTEGER: copy
```

4. Pour que soit visible le propriétaire actuel de l'autorisation de modification, vous pouvez définir le propriétaire lorsque vous saisissez la commande **vtpVlanEditBufferOwner**.

```
snmpset -c private crumpy vtpVlanEditBufferOwner.1 octetstring "Gerald"
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControlEntry.vtpVlanEditBufferOwner.1 : OCTET STRING- (ascii): Gerald
```

5. Cet exemple montre comment vérifier que la table existe :

```
snmpwalk -c public crumpy vtpVlanEditTable
vtpVlanEditState.1.1 : INTEGER: operational
vtpVlanEditState.1.2 : INTEGER: operational
vtpVlanEditState.1.3 : INTEGER: operational
..
```

6. Cet exemple est VLAN 11 et vous montre comment créer une ligne et définir le type et le nom :

```
snmpset -c private crumpy vtpVlanEditRowStatus.1.11 integer 4
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditEntry.vtpVlanEditRowStatus.1.11 : INTEGER: createAndGo

snmpset -c private crumpy vtpVlanEditType.1.11 integer 1
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditEntry.vtpVlanEditType.1.11 : INTEGER: ethernet

snmpset -c private crumpy vtpVlanEditName.1.11 octetstring "test_11_gerald"
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditEntry.vtpVlanEditName.1.11 : DISPLAY STRING- (ascii): test_11_gerald
```

7. Définissez **vtpVlanEditDot10Said**. Il s'agit du numéro de VLAN + 100000 converti en hexadécimal. Cet exemple crée le réseau VLAN 11, donc **vtpVlanEditDot10Said** devrait être : $11 + 100000 = 100011 \rightarrow$ Hex : 000186AB

```
snmpset -c private crumpy vtpVlanEditDot10Said.1.11 octetstringhex 000186AB
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditDot10Said.1.11 : OCTET STRING- (hex): length = 4
0: 00 01 86 ab -- -- -- -- -- -- -- -- -- -- .....
```

8. Une fois que vous avez créé le réseau VLAN 11, vous devez appliquer les modifications. Utilisez de nouveau l'OID `vtpVlanEditOperation`. Cliquez cette fois sur le bouton **Apply** [appliquer] pour confirmer les paramètres :

```
snmpset -c private crumpy vtpVlanEditOperation.1 integer 3
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControlEntry.vtpVlanEditOperation.1 : INTEGER: apply
```

9. Vérifiez que le réseau VLAN a été créé avec succès. Utilisez l'OID `vtpVlanApplyStatus`.

Vérifiez le processus jusqu'à ce que l'état indique `succeeded` :

```
snmpget -c public crumpy vtpVlanApplyStatus.1
vtpVlanApplyStatus.1 : INTEGER: inProgress
snmpget -c public crumpy vtpVlanApplyStatus.1
vtpVlanApplyStatus.1 : INTEGER: inProgress
snmpget -c public crumpy vtpVlanApplyStatus.1
vtpVlanApplyStatus.1 : INTEGER: succeeded
```

10. La dernière action consiste à valider les modifications et à octroyer les autorisations afin que les autres utilisateurs puissent, dans leur système NMS, ajouter, modifier ou supprimer des réseaux VLAN.

```
snmpset -c private crumpy vtpVlanEditOperation.1 integer 4
vtpVlanEditOperation.1 : INTEGER: release
```

11. Vérifiez que la mémoire tampon est vide :

```
snmpwalk -c public crumpy vtpVlanEditTable
no MIB objects contained under subtree.
```

12. Vérifiez que le réseau VLAN 11 a été créé sur le commutateur au moyen de la commande CLI `show vlan` ou d'un `snmpwalk` :

```
snmpwalk -c public crumpy vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.1 : INTEGER: operational
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.11 : INTEGER: operational
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.48 : INTEGER: operational
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.1002 : INTEGER: operational
...
```

[Ajouter un réseau VLAN à un commutateur Cisco Catalyst avec SNMP](#)

[Instructions en une étape](#)

Le processus en une étape utilise les numéros OID au lieu des noms OID comme le processus étape par étape précédent. Consultez les [détails de la MIB](#) pour la traduction. Cet exemple illustre la création du réseau VLAN 6 :

```
snmpset -c private crumpy 1.3.6.1.4.1.9.9.46.1.4.1.1.1.1 integer 2
```

```
1.3.6.1.4.1.9.9.46.1.4.1.1.3.1 octetstring "gcober"
```

```
snmpset -c private gooroo 1.3.6.1.4.1.9.9.46.1.4.2.1.11.1.6 integer 4
```

```
1.3.6.1.4.1.9.9.46.1.4.2.1.3.1.6 integer 1 1.3.6.1.4.1.9.9.46.1.4.2.1.4.1.6 octetstring "vlan6"
```

```
1.3.6.1.4.1.9.9.46.1.4.2.1.6.1.6 octetstringhex 000186A6 1.3.6.1.4.1.9.9.46.1.4.1.1.1.1 integer 3
```

```
snmpset -c private gooroo 1.3.6.1.4.1.9.9.46.1.4.1.1.1.1 integer 4
```

```
snmpwalk -c public crumpy 1.3.6.1.4.1.9.9.46.1.3.1.1.2
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.1 :  
INTEGER: operational
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.6 :  
INTEGER: operational
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1.11 :  
INTEGER: operational
```

Remarque : Certaines versions de SNMP exigent l'utilisation d'un (.) avant l'OID dans les commandes SNMP SET.

[Supprimer un réseau VLAN d'un commutateur Cisco Catalyst avec SNMP](#)

[Step-by-Step Instructions](#)

Dans cet exemple, le réseau VLAN 48 est supprimé du commutateur. Consultez la section [Ajouter un réseau VLAN à un commutateur Cisco Catalyst avec SNMP](#) pour en savoir plus. La différence entre la section où vous supprimez un réseau VLAN et celle où vous ajoutez un réseau VLAN est que vous utilisez la commande **destroy** au lieu de la commande **CreateAndGo** pour **vtpVlanEditRowStatus** :

1. Saisissez la commande pour supprimer le réseau VLAN 48 :

```
snmpset -c private crumpy vtpVlanEditOperation.1 integer 2
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControlEntry.  
vtpVlanEditOperation.1 : INTEGER: copy
```

```
snmpset -c private crumpy vtpVlanEditRowStatus.1.48 integer 6
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditEntry.vtpVla  
nEditRowStatus.1.48 : INTEGER: destroy
```

2. Pour vérifier que le réseau VLAN 48 a été supprimé, utilisez **vtpVlanState** ou **show vlan** sur la CLI :

```
snmpwalk -c public crumpy vtpVlanState
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1  
.1 : INTEGER: operational
```

```
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState.1  
.1002 : INTEGER: operational
```

```
...
```

[Ajouter un port à un réseau VLAN sur un commutateur Cisco Catalyst avec SNMP](#)

Cet exemple montre comment ajouter un port Fast Ethernet 0/5 au VLAN 48.

1. Pour vérifier quel ifIndex a le Fast Eth 0/5, saisissez la commande **snmpwalk** d'**ifDescr** :

```
snmpwalk -c public crumpy ifDescr
...
interfaces.ifTable.ifEntry.ifDescr.6 : DISPLAY STRING- (ascii): FastEthernet0/5
...
```

2. Comme vous savez que le port Fast Eth 0/5 a un ifIndex de 6, ajoutez le port au VLAN 48 :

```
snmpset -c private crumpy vmVlan.6 integer 48
cisco.ciscoMgmt.ciscoVlanMembershipMIB.ciscoVlanMembershipMIBObjects.vmMembership.vmMembers
hipTable.vmMembershipEntry.vmVlan.6 : INTEGER: 48
```

3. Vérifiez que le port a été correctement ajouté en interrogeant de nouveau le même OID.

```
snmpget -c public crumpy vmVlan.6
cisco.ciscoMgmt.ciscoVlanMembershipMIB.ciscoVlanMembershipMIBObjects.vmMembership.vmMembers
hipTable.vmMembershipEntry.vmVlan.6 : INTEGER: 48
```

Vous pouvez également vérifier ce qui suit sur le commutateur :

```
crumpy#sh vlan
```

VLAN Name	Status	Ports
1 default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4, Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10, Fa0/11, Fa0/12, Fa0/13, Fa0/14, Fa0/15, Fa0/16, Fa0/17, Fa0/18, Fa0/19, Fa0/20, Fa0/21, Fa0/22, Fa0/23, Fa0/24, Gi0/1, Gi0/2
48 VLAN0048	active	Fa0/5

[Comment changer un port d'un réseau VLAN à un autre](#)

Cet exemple montre comment le port Fast Eth 0/3 appartient au VLAN 48 et comment le déplacer vers le VLAN 1 (VLAN par défaut) :

1. Pour vérifier quel ifIndex a le Fast Eth 0/3, saisissez la commande **snmpwalk** de l'**ifDescr** :

```
snmpwalk -c public crumpy ifDescr
...
interfaces.ifTable.ifEntry.ifDescr.4 : DISPLAY STRING- (ascii): FastEthernet0/3
...
```

2. Comme vous savez que le port Fast Eth 0/3 a un ifIndex de 4, vous pouvez vérifier à quel VLAN appartient le port actuellement :

```
snmpget -c public crumpy vmVlan.4
cisco.ciscoMgmt.ciscoVlanMembershipMIB.ciscoVlanMembershipMIBObjects.vmMembership.vmMembers
hipTable.vmMembershipEntry.vmVlan.4 : INTEGER: 48
```

3. Le port appartient au réseau VLAN 48.

```
snmpset -c private crumpy vmVlan.4 integer 1
cisco.ciscoMgmt.ciscoVlanMembershipMIB.ciscoVlanMembershipMIBObjects.vmMembership.vmMembers
hipTable.vmMembershipEntry.vmVlan.4 : INTEGER: 1
```

4. Pour déplacer le port du réseau VLAN 48 au réseau VLAN 1, saisissez la commande

snmpset de vmVlan.

5. Pour vérifier si le port a été remplacé par un autre réseau VLAN, interrogez de nouveau vmVlan :

```
snmpget -c public crumpy vmVlan.4
```

```
cisco.ciscoMgmt.ciscoVlanMembershipMIB.ciscoVlanMembershipMIBObjects.vmMembership.vmMembers  
hipTable.vmMembershipEntry.vmVlan.4 : INTEGER: 1
```

Vous pouvez également vérifier ceci sur le commutateur :Avant la modification :

```
crumpy#sh vlan
```

VLAN Name	Status	Ports
1 default	active	Fa0/1, Fa0/2, Fa0/4, Fa0/5, Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10, Fa0/11, Fa0/12, Fa0/13, Fa0/14, Fa0/15, Fa0/16, Fa0/17, Fa0/18, Fa0/19, Fa0/20, Fa0/21, Fa0/22, Fa0/23, Fa0/24, Gi0/1, Gi0/2
48 VLAN0048	active	Fa0/3

Après la modification :

```
crumpy#sh vlan
```

VLAN Name	Status	Ports
1 default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4, Fa0/5, Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10, Fa0/11, Fa0/12, Fa0/13, Fa0/14, Fa0/15, Fa0/16, Fa0/17, Fa0/18, Fa0/19, Fa0/20, Fa0/21, Fa0/22, Fa0/23, Fa0/24, Gi0/1, Gi0/2
48 VLAN0048	active	

Remarque : Vous pouvez apporter d'autres modifications notamment au nom du VLAN ou au propriétaire. Consultez l'ensemble de la MIB pour en savoir plus sur l'OID.

[Informations connexes](#)

- [Support technique - Cisco Systems](#)