

How To Add, Modify, and Remove VLANs on a Catalyst Using SNMP

Document ID: 45080

Introduction

Prerequisites

Requirements

Components

Conventions

Background

Details of the MIB Variables Including Object Identifiers (OIDs)

Add a VLAN to a Cisco Catalyst Switch With SNMP

Step-by-Step Instructions

Add a VLAN to a Cisco Catalyst Switch With SNMP

One Step Instructions

Delete a VLAN from a Cisco Catalyst Switch With SNMP

Step-by-Step Instructions

Add a Port to a VLAN on a Cisco Catalyst Switch With SNMP

How to Change a Port from One VLAN to Another VLAN

Related Information

Introduction

This document describes how to create and delete VLANs on a Cisco Catalyst switch that uses Simple Network Management Protocol (SNMP). It also describes how to add ports to a VLAN with SNMP.

Prerequisites

Requirements

Before you use the information in this document, ensure that you understand:

- How the `ifTable` and `ifIndexes` work
- How VLANs work on Cisco Catalyst switches
- How to view VLAN information on Cisco Catalysts switches
- The general use of SNMP `get`, `set`, and `walk` commands

Components

This document is for Catalyst switches that run regular Catalyst OS or Catalyst IOS that support the IF-MIB, CISCO-VTP-MIB and CISCO-VLAN-MEMBERSHIP-MIB. The information in this document is based on these software and hardware versions:

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

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

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Background

Details of the MIB Variables Including Object Identifiers (OIDs)

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 instantiate 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 instanciate
 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 }
```

Add a VLAN to a Cisco Catalyst Switch With SNMP

Step-by-Step Instructions

In the example shown below, VLAN 11 is added to the switch:

1. In order to check which VLANs are currently configured on the switch, issue an **snmpwalk** on the **vtpVlanState** OID:

Note: The last number in the OID is the VLAN number.

```
snmpwalk -c public crumpy vtpVlanState  
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlan  
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlan  
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlan
```

2. Verify if the edition is in use by another NMS station or device. The edition is not in use if you see this message: no MIB objects contained under subtree:

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

3. The edition is not in use, so it is safe to start to edit. Set the **vtpVlanEditOperation** to the copy state (integer 2). This allows you to create the VLAN.

```
snmpset -c private crumpy vtpVlanEditOperation.1 integer 2  
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControl
```

4. In order to make the current owner of the edit permission visible, you can set the owner when you issue the command, **vtpVlanEditBufferOwner**.

```
snmpset -c private crumpy vtpVlanEditBufferOwner.1 octetstring "Gerald"  
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpEditControlTable.vtpEditControl
```

5. This example shows how to verify that the table exists:

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

6. This example is VLAN 11 and shows you how to create a row and set the type and the name:

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

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

snmpset -c private crumpy vtpVlanEditName.1.11 octetstring "test_11_gerald"
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanEdit.vtpVlanEditTable.vtpVlanEditEntry
```

7. Set the **vtpVlanEditDot10Said**. This is the VLAN number + 100000 translated to hexadecimal. This example creates VLAN 11, so the **vtpVlanEditDot10Said** should be: $11 + 100000 = 100011 \rightarrow$ Hex: 000186AB

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

8. When you have created VLAN 11, you must apply the modifications. Use the **vtpVlanEditOperation** OID again. This time use the **Apply** to confirm the settings :

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

9. Verify that the VLAN was created successfully. Use the OID **vtpVlanApplyStatus**. Check the process until the status reads: 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. The last action is to commit the modifications and release the permissions so that other users can add, modify, or delete VLANs from their NMS.

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

11. Verify that the buffer is empty:

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

12. Verify that VLAN 11 was created on the switch with the CLI command **show vlan** or with an **snmpwalk**:

```
snmpwalk -c public crumpy vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState
cisco.ciscoMgmt.ciscoVtpMIB.vtpMIBObjects.vlanInfo.vtpVlanTable.vtpVlanEntry.vtpVlanState
&
```



```

48    VLAN0048    active    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
                                           Fa0/3

```

After the change:

```

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

```

Note: You can make other changes, such as the VLAN name, the owner, and much more. Refer to the entire MIB for more details on OID.

Related Information

- [Technical Support – Cisco Systems](#)

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Oct 26, 2005

Document ID: 45080
