- Virtual LANs (VLANs) and VTP -

Collision vs. Broadcast Domains

A **collision domain** is simply defined as any physical segment where a **collision** can occur. Hubs can only operate at half-duplex, and thus all ports on a hub belong to the same collision domain.

Layer-2 switches can operate at full duplex. *Each individual port* on a switch belongs to its *own collision domain*. Thus, Layer-2 switches create **more collision domains**, which results in **fewer collisions.**

Like hubs though, Layer-2 switches belong to only *one* **broadcast domain.** A Layer-2 switch will forward both broadcasts and multicasts out *every* port but the originating port.

Only Layer-3 devices separate broadcast domains. Because of this, Layer-2 switches are poorly suited for large, scalable networks. The Layer-2 header provides no mechanism to differentiate one *network* from another, only one *host* from another.

Virtual LANs (VLANs)

By default, a switch will forward both broadcasts and multicasts out *every* port but the originating port. However, a switch can be *logically* segmented into separate broadcast domains, using **Virtual LANs** (or **VLANs**).

Each VLAN represents a unique broadcast domain:

- Traffic between devices within the *same* VLAN is switched.
- Traffic between devices in *different* VLANs requires a Layer-3 device to communicate.

Broadcasts from one VLAN will not be forwarded to another VLAN. The logical separation provided by VLANs is **not a Layer-3 function.** VLAN tags are inserted into the **Layer-2 header**.

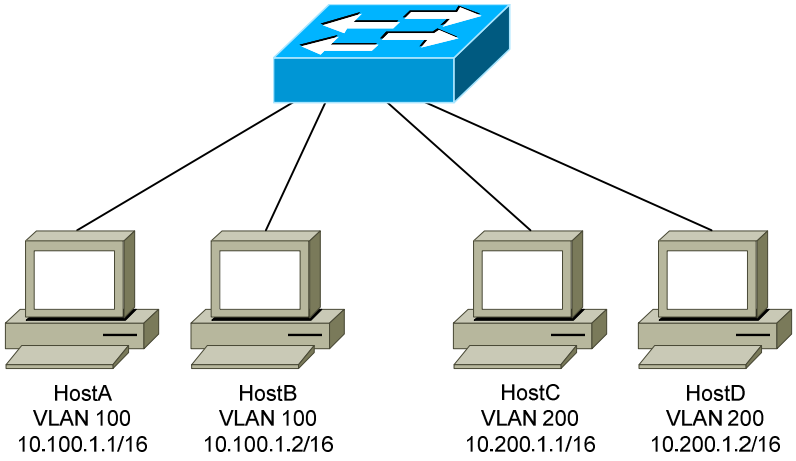
Thus, a switch that supports VLANs is not necessarily a Layer-3 switch. However, a purely Layer-2 switch cannot route between VLANs.

Remember, though VLANs provide separation for *Layer-3* broadcast domains, they are still a *Layer-2* function. A VLAN often has a direct relationship with an IP subnet, though this is not a requirement.

* * *

VLAN Example

Consider the following example:



Four hosts are connected to a Layer-2 switch that supports VLANs:

- HostA and HostB belong to VLAN 100
- HostC and HostD belong to VLAN 200

Because HostA and HostB belong to the *same* VLAN, they belong to the same broadcast domain as well. The switch will be able to forward frames between the two hosts without the need of a Layer-3 device, such as a router.

Likewise, HostC and HostD belong to the same VLAN, and thus the same broadcast domain. They also will be able to communicate without an intervening Layer-3 device.

However, HostA and HostB *will not* be able to communicate with HostC and HostD. They are members of separate VLANs, and belong in *different* broadcast domains. Thus, a Layer-3 device is required for those hosts to communicate.

A broadcast sent from a host in VLAN 100 will be received by all other hosts in that same VLAN. However, that broadcast will not be forwarded to any other VLAN, such as VLAN 200.

On Cisco switches, all interfaces belong to **VLAN 1** by default**.** VLAN 1 is also considered the **Management VLAN,** and should be dedicated for system traffic such as CDP, STP, VTP, and DTP.

* * *

Advantages of VLANs

VLANs provide the several benefits:

- **Broadcast Control –** eliminates unnecessary broadcast traffic, improving network performance and scalability.
- **Security –** logically separates users and departments, allowing administrators to implement access-lists to control traffic between VLANs.
- **Flexibility –** removes the physical boundaries of a network, allowing a user or device to exist anywhere.

VLANs are very common in LAN and campus networks. For example, user networks are often separated from server networks using VLANs.

VLANs can span across WANs as well, though there are only limited scenarios where this is necessary or recommended.

VLAN Membership

VLAN membership can be configured one of two ways:

- **Statically**
- **Dynamically**

Statically assigning a VLAN involves manually assigning an individual or group of ports to a VLAN. Any host connected to that port (or ports) immediately becomes a member of that VLAN. This is *transparent* to the host - it is unaware that it belongs to a VLAN.

VLANs can be assigned **dynamically** based on the MAC address of the host. This allows a host to remain in the same VLAN, regardless of which switch port it is connected to.

Dynamic VLAN assignment requires a separate database to maintain the MAC-address-to-VLAN relationship. Cisco developed the **VLAN Membership Policy Server (VMPS)** to provide this functionality.

In more sophisticated systems, a user's network account can be used to determine VLAN membership, instead of a host's MAC address.

Static VLAN assignment is far more common than dynamic, and will be the focus of this guide.

* * *

Creating VLANs

By default, all interfaces belong to **VLAN 1.** To assign an interface to a different VLAN, that VLAN must first be *created*:

Switch(config)# *vlan 100*
Switch(config-vlan)# *name SERVERS*

The first command creates VLAN 100, and enters VLAN configuration mode. The second command assigns the name *SERVERS* to this VLAN.

Note that naming a VLAN is *not* required.

The standard range of VLAN numbers is **1 – 1005,** with VLANs 1002-1005 reserved for legacy Token Ring and FDDI purposes.

A switch operating in VTP **transparent mode** can *additionally* use the VLAN range of **1006 – 4094.** These are known as extended-range VLANs. VTP is covered in great detail later in this guide.

To remove an individual VLAN:

Switch(config)# *no vlan 100*

Note that VLAN 1 cannot be removed. To remove a group of VLANs:

Switch(config)# *no vlan 150-200*

To view all created VLANs, including the interfaces assigned to each VLAN:

Switch# *show vlan*

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------
1    default                          active    gi1/1-24
100  SERVERS                          active
1002 fddi-default                     suspended
1003 token-ring-default               suspended
1004 fddinet-default                  suspended
1005 trnet-default                    suspended
```

Note that no interfaces have been assigned to the newly created VLAN 100 yet.

* * *

Statically Assigning VLANs

To statically assign an interface into a specific VLAN:

Switch(config)# *interface gi1/10*
Switch(config-if)# *switchport mode access*
Switch(config-if)# *switchport access vlan 100*

The first command enters interface configuration mode. The second command indicates that this is an *access* port, as opposed to a *trunk* port. This will be explained in detail shortly.

The third command assigns this access port to VLAN *100*. Note that the VLAN *number* is specified, and not the VLAN *name*.

The *show vlan* command should now reflect the new VLAN assignment:

Switch# *show vlan*

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------
1    default                          active    gi1/1-9,11-24
100  SERVERS                          active    gi1/10
1002 fddi-default                     suspended
1003 token-ring-default               suspended
1004 fddinet-default                  suspended
1005 trnet-default                    suspended
```

For switches running in VTP **server** or **client mode**, the *list* of VLANs are stored in a database file named **vlan.dat.** The vlan.dat file is usually stored in **flash**, though on some switch models it is stored in NVRAM. The VLAN database will be maintained even if the switch is rebooted.

For switches running in VTP **transparent mode**, the list of VLANs is stored in the startup-config file in NVRAM. VTP is covered extensively later in this guide.

Regardless of VTP mode, the VLAN *assignment* for every switch interface is stored in the switch's **startup-config**.

* * *

VLAN Port Types

A VLAN-enabled switch supports two types of ports:

- **Access ports**
- **Trunk ports**

An **access port** is a member of only a *single* VLAN. Access ports are most often used to connect host devices, such as computers and printers. By default on Cisco switches, *all* switch ports are access ports.

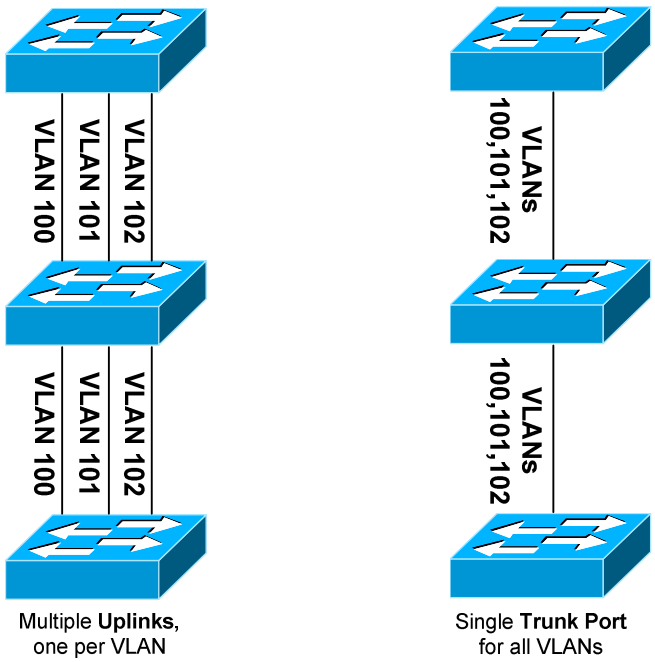
Any host connected to an access port immediately becomes a member of the VLAN configured on that port. This is *transparent* to the host - it is unaware that it belongs to a VLAN.

It is possible for a VLAN to span more than one switch. There are two methods of connecting a VLAN across multiple switches:

- Create *uplink* access ports between the switches, one for each VLAN.
- Create a **trunk connection** between the switches.

A **trunk port** is not a member of a single VLAN. The traffic from *any or all* VLANs can traverse trunk links to reach other switches.

Uplinking access ports quickly becomes unfeasible in large switching environments. The following illustrates the advantage of using trunk ports:

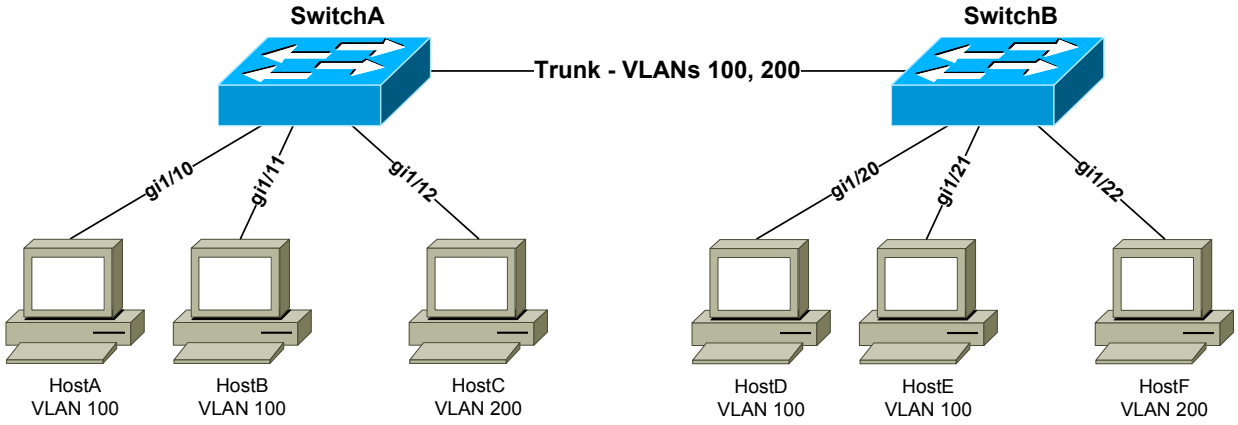


* * *

VLAN Frame-Tagging

When VLANs span multiple switches, a mechanism is required to identify which VLAN a frame belongs to. This is accomplished through **frame tagging,** which places a VLAN ID in each frame.

Tagging *only* occurs when a frame is **sent out a trunk port**. Traffic sent out access ports is never tagged. Consider the following example:



If HostA sends a frame to HostB, no frame tagging will occur:

- The frame never leaves the SwitchA.
- The frame stays within its own VLAN.
- The frame is simply **switched** to HostB.

If HostA sends a frame to HostC, which is in a separate VLAN:

- The frame *again* never leaves the switch.
- Frame tagging will *still* not occur.
- Because HostC is in a different VLAN, the frame must be **routed**.

If HostA sends a frame to HostD, which is on a separate switch:

- The frame is sent out the trunk port to SwitchB.
- The frame *must* be **tagged** as it is sent out the trunk port.
- The frame is tagged with its VLAN ID - VLAN 100 in this example.
- When SwitchB receives the frame, it will only forward it out ports belonging to VLAN 100 – *gi1/20* and *gi1/21*.
- If SwitchB has HostD's MAC address in its table, it will forward the frame only out the appropriate port – *gi1/20*.
- The VLAN tag is stripped from the frame before being forwarded to the host.

* * *

Frame Tagging Protocols

Cisco switches support two frame tagging protocols:

- **Inter-Switch Link (ISL)**
- **IEEE 802.1Q**

The tagging protocol can be manually specified on a trunk port, or dynamically negotiated using Cisco's proprietary **Dynamic Trunking Protocol (DTP)**.

Inter-Switch Link (ISL)

Inter-Switch Link (ISL) is Cisco's proprietary frame tagging protocol. ISL supports several technologies:

- Ethernet
- Token Ring
- FDDI
- ATM

ISL *encapsulates* a frame with an additional header (**26 bytes**) and trailer (**4 bytes**). Thus, ISL increases the size of a frame by **30 bytes**.

The header contains several fields, including a 15-bit VLAN ID. The trailer contains an additional 4-byte CRC to verify data integrity.

Normally, the maximum possible size of an Ethernet frame is **1518 bytes.** This is known as the Maximum Transmission Unit (**MTU**). Most Ethernet devices use a *default* MTU of **1514 bytes.**

ISL increases the frame size by another 30 bytes. Thus, most switches will disregard ISL-tagged frames as being **oversized,** and drop the frame. An oversized frame is usually referred to as a **giant**. Somewhat endearingly, a *slightly* oversized frame is known as a **baby giant.**

Cisco switches are specifically engineered to support these giant ISL – tagged frames. Note that this is a key reason why ISL is Cisco-proprietary.

ISL supports a maximum of **1000 VLANs** on a trunk port. ISL is also almost entirely deprecated - most modern Cisco switches no longer support it.

(Reference: http://www.cisco.com/c/en/us/support/docs/lan-switching/8021q/17056-741-4.html)

* * *

IEEE 802.1Q

IEEE 802.1Q, otherwise referred to as **dot1Q**, is an industry-standard frame-tagging protocol.

802.1Q is supported by nearly all switch manufacturers, including Cisco. Because 802.1Q is an open standard, switches from different vendors can be trunked together.

Recall that ISL encapsulates a frame with an additional header and trailer. In contrast, 802.1Q *embeds* a **4-byte VLAN tag** directly into the Layer-2 frame header. Because the Layer-2 header is modified, 802.1Q must recalculate the frame's CRC value.

The VLAN tag includes a 12-bit VLAN ID. This tag increases the size of an Ethernet frame, from its default of 1514 bytes to 1518 bytes. Nearly all modern switches support the 802.1Q tag and the slight increase in frame size.

802.1Q supports a maximum of **4096 VLANs** on a trunk port.

Configuring Trunk Links

To manually configure an interface as a trunk port:

Switch(config)# *interface gi2/24*
Switch(config-if)# *switchport mode trunk*

For a switch that supports both ISL and 802.1Q, the tagging or *encapsulation* protocol must be configured first:

Switch(config)# *interface gi2/24*
Switch(config-if)# *switchport trunk encapsulation isl*
Switch(config-if)# *switchport mode trunk*

Switch(config)# *interface gi2/24*
Switch(config-if)# *switchport trunk encapsulation dot1q*
Switch(config-if)# *switchport mode trunk*

Important note: Both sides of the trunk must be configured with the *same* tagging protocol. Otherwise, a trunk connection will not form.

If the switch only supports 802.1Q, the *switchport trunk encapsulation* command will not be available.

* * *

Configuring Trunk Links (continued)

The switch can *negotiate* the tagging protocol, using DTP:

```
Switch(config)#  interface gi2/24
Switch(config-if)#  switchport trunk encapsulation negotiate
Switch(config-if)#  switchport mode trunk
```

The tagging protocol that is supported by *both* switches will be used. If the switches support both ISL and 802.1Q, **ISL** will be the preferred protocol.

By default, **all active VLANs** are allowed to traverse a trunk link. While this is convenient, a good security practice is to allow only necessary VLANs over a trunk.

To explicitly *allow* a subset of VLANs on a trunk port:

```
Switch(config)#  interface gi2/24
Switch(config-if)#  switchport trunk allowed vlan 3,9,11-15
```

The above command will force the trunk link to only forward traffic from VLANS *3, 9,* and *11 – 15.* To *remove* a VLAN from the allowed list:

```
Switch(config)#  interface gi2/24
Switch(config-if)#  switchport trunk allowed vlan remove 12
```

To *add* a specific VLAN back into the allowed list:

```
Switch(config)#  interface gi2/24
Switch(config-if)#  switchport trunk allowed vlan add 25
```

Important Note: It is common to restrict the allowed VLANs on a trunk link, and then add to the allowed list as new VLANs are created. **However**, don't forget to use the *add* parameter. If *add* is omitted, the command will *replace* the list of allowed VLANs on the trunk link, to the great distress of network admins everywhere (sorry Karl). Always remember to *add*! ☺

To allow all VLANs *except* for a specific range:

```
Switch(config-if)#  switchport trunk allowed vlan except 50-99
```

To allow *all* VLANs again:

```
Switch(config-if)#  switchport trunk allowed vlan all
```

* * *

Native VLANs

Recall that a trunk port *tags* frames with a VLAN ID. But what happens if a trunk port receives an *untagged* frame?

The **native VLAN** determines the VLAN that untagged traffic belongs to. By default on all trunking ports, the native VLAN is **VLAN 1**. The native VLAN can be changed on a per trunk port basis:

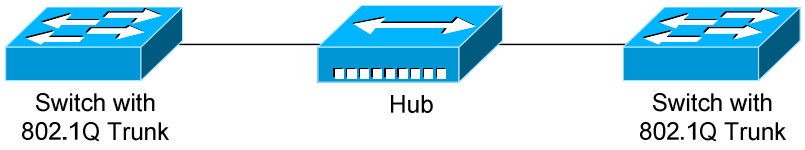
```
Switch(config)#  interface gi2/24
Switch(config-if)#  switchport mode trunk
Switch(config-if)#  switchport trunk native vlan 42
```

Only one native VLAN can be assigned to a trunk port. All untagged traffic *received* on this port will become a member of the native VLAN. Additionally, frames belonging to the native VLAN are not tagged when being *sent* out a trunk port.

Native VLANS are only supported on **802.1Q** trunk ports**.** ISL does not support untagged frames, and will *always* tag frames from *all* VLANs.

The native VLAN must be configured **identically on both sides of the 802.1Q trunk,** otherwise the switches will not form a trunk connection.

The original intent of native VLANs was for legacy compatibility with hubs. Consider the following deprecated example:



The hub has no knowledge of VLANs or 802.1Q. Traffic from hosts connected to the hub will be forwarded to the switches untagged, which in turn will place the untagged traffic into the native VLAN.

Native VLANs pose a **security risk**, allowing an attacker to *hop* to another VLAN by *double-tagging* a frame. This can be mitigated by changing the native VLAN to an unused or disabled VLAN. A better solution is to force trunk ports to tag native VLAN traffic - globally or on a per-trunk basis:

```
Switch(config)#  vlan dot1q tag native

Switch(config)#  interface gi2/24
Switch(config-if)#  switchport trunk native vlan tag
```

* * *

Dynamic Trunking Protocol (DTP)

Recall that a trunk's frame tagging protocol can be autonegotiated, through the use of the **Dynamic Trunking Protocol (DTP)**.

DTP can also negotiate **whether a port becomes a trunk at all**. Previous examples demonstrated how to manually configure a port to trunk:

```
Switch(config)# interface gi2/24
Switch(config-if)# switchport mode trunk
```

DTP has two modes to *dynamically* decide whether a port becomes a trunk:

- **Desirable** – the port will *actively* attempt to form a trunk with the remote switch. This is the default setting.
- **Auto** – the port will *passively* wait for the remote switch to initiate the trunk.

To configure the DTP mode on an interface:

```
Switch(config)# interface gi2/24
Switch(config-if)# switchport mode dynamic desirable
Switch(config-if)# switchport mode dynamic auto
```

Trunk ports send out DTP frames **every 30 seconds** to indicate their configured mode.

A trunk ***will form*** in the following configurations:

- manual trunk ←→ manual trunk
- manual trunk ← → dynamic desirable
- manual trunk ← → dynamic auto
- dynamic desirable ← → dynamic desirable
- dynamic desirable ← → dynamic auto

A trunk ***will never form*** if the two sides of the trunk are set to dynamic auto, as both ports are waiting for the other to initialize the trunk.

It is best practice to manually configure trunk ports, to avoid DTP negotiation errors. DTP is also vulnerable to VLAN spoofing attacks.

To explicitly disable DTP:

```
Switch(config)# interface gi2/24
Switch(config-if)# switchport mode trunk
Switch(config-if)# switchport nonegotiate
```

* * *

Troubleshooting Trunk Connections

A trunk connection requires several parameters to be configured identically on both sides of the trunk:

- **Trunk Mode**
- **Frame-tagging protocol**
- **Native VLAN**
- **Allowed VLANs**
- **VTP Domain** – only when using DTP to negotiate a trunk

If there is a mismatch in configuration, the trunk connection will never become active.

To determine whether an interface is an access or trunk port:

Switch# *show interface gi2/24 switchport*

```
Name: Gi2/24
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 42

<snip>
```

To view the status of all trunk links:

Switch# *show interface trunk*

```
Port        Mode         Encapsulation      Status       Native VLAN
Fa0/24      on           802.1q             trunking     42

Port        Vlans allowed on trunk
Fa0/24      3,9,11-15

Port        Vlans allowed and active in management domain
Fa0/24      3,9

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/24      3,9
```

Note that VLANs *11-15* are not active. Most likely, no interfaces have been assigned to those VLANs.

If there are no interfaces in an active trunking state, the *show interface trunk* command will return no output.

* * *

VLAN Trunking Protocol (VTP)

Maintaining a consistent VLAN database can be difficult in a large switching environment.

Cisco's proprietary **VLAN Trunking Protocol (VTP)** simplifies this management - updates to the VLAN database are propagated to all switches using **VTP advertisements.**

VTP requires that all participating switches join a **VTP domain**. Switches must belong to the same domain to share VLAN information, and a switch can only belong to a single domain.

VTP Versions

There are *three* versions of VTP. **VTP version 1** supports the standard 1 – 1005 VLAN range. VTP version 1 is also default on Catalyst switches.

VTP version 2 introduces some additional features:

- Token Ring support
- VLAN consistency checks
- Domain-independent transparent pass through

VTPv1 and v2 are **not compatible.** The VTP version is dictated by the **VTP server**, discussed in detail shortly. If the VTP server is configured for VTPv2, all other switches in the VTP domain will change to v2 as well.

Until recently, **VTP Version 3** was supported on only limited Cisco switch platforms. VTPv3 was built to be flexible, and can forward both VLAN and other database information, such as Multiple Spanning Tree (MST) protocol.

Other enhancements provided by VTPv3 include:

- Support for the extended 1006-4094 VLAN range.
- Support for private VLANs.
- Improved VTP authentication.
- Protection from accidental database overwrites, by using VTP primary and secondary servers.
- Ability to enable VTP on a per-port basis.

(Reference: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3560/software/release/12-2_52_se/configuration/guide/3560scg/swvtp.html)

* * *

VTP Modes

A switch using VTP must operate in one of three **modes**:

- **Server**
- **Client**
- **Transparent**

VTP servers are responsible for creating, deleting, or modifying entries in the VLAN database. Each VTP domain must have at least one VTP server, and this is the **default mode** for Cisco switches.

Servers advertise the VLAN database to all other switches in the VTP domain, including other servers. VTP servers can only advertise the standard 1-1005 VLAN range, and advertisements are only sent out **trunk** ports.

VTP clients cannot modify the VLAN database, and rely on advertisements from other switches to update VLAN information. A client will also *forward* VTP advertisements out every trunk port.

Remember: switches must be in the **same VTP Domain** to share and accept updates to the VLAN database. Only servers can change the VLAN database.

A **VTP transparent** switch maintains its own local VLAN database, and does not directly participate in the VTP domain. A transparent switch will never accept VLAN database information from another switch, even a server. Also, a transparent switch will never advertise its local VLAN database to another switch.

Transparent switches will **pass through** advertisements from other switches in the VTP domain. The VTP version dictates how the pass through is handled:

- **VTP version 1** – the transparent switch will only pass through advertisements from the *same* VTP domain.
- **VTP version 2** – the transparent switch will pass through advertisements from *any* VTP domain.

* * *

VTP Advertisements – Revision Number

Recall that updates to the VLAN database are propagated using **VTP advertisements.** VTP advertisements are always sent out **trunk ports**, on VLAN 1.

VTP advertisements are marked with a 32-bit **configuration revision number**, to identify the most current VLAN database revision. Any change to the VLAN database increments the configuration revision number by 1. Thus, a *higher* number represents a *newer* database revision.

A switch will only accept an advertisement if the revision number is *higher* than the current VLAN database. Advertisements with a *lower* revision number are ignored.

Important note: While only VTP servers can *change* the VLAN database, VTP clients can *advertise* updates, to other clients and even to a server! As long as the revision number is higher, the switch will accept the update.

This can result in a newly-introduced switch advertising a *blank* or *incorrect* VLAN database to all other switches in the domain. Switch ports would then lose their VLAN memberships, resulting in a significant network outage.

This can be avoided when implementing a new switch into the VTP domain. Best practice is to configure a new switch as a VTP client, and reset its revision number to **zero** before deploying into a production network.

There are two methods of resetting the revision number to zero on a switch:

1. Change the VTP domain name, and then change it back to the original name.
2. Change the VTP mode to transparent, and then change it back to either server or client. Transparent switches always a revision number of 0.

VTP has fallen out of favor, due to the risk of an unintentional overwrite of the VLAN database. Until very recently, Cisco did not support VTP on the Nexus platform of switches.

VTPv3 directly addresses this risk through the use of VTP **primary** and **secondary** servers. Only the primary server is allowed to update the VLAN database on other switches. Only one primary server is allowed per domain.

(Reference: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/solution_guide_c78_508010.html)

* * *

VTP Advertisements – Message Types

Three **message types** exist for VTP advertisements:

- **Summary Advertisement**
- **Subset Advertisement**
- **Advertisement Request**

Both VTP servers and clients will send out a **summary advertisement** every **300 seconds**. Summary advertisements contain the following information about the VTP domain:

- VTP version
- Domain name
- Configuration revision number
- Time stamp
- MD5 digest

Summary advertisements are also sent when a **change occurs** to the VLAN database. The summary is then followed with a **subset advertisement**, which actually contains the full, updated VLAN database.

A subset advertisement will contain the following information:

- VTP version
- Domain name
- Configuration revision number
- VLAN IDs for each VLAN in the database
- VLAN-specific information, such as the VLAN name and MTU

Important note: Switches will only accept summary and subset advertisements if the *domain name* and *MD5 digest* match. Otherwise, the advertisements are ignored.

If a switch receives a summary advertisement with a revision number *higher* than its own, it will send out an **advertisement request.** VTP servers will then respond with an updated summary and subset advertisement so that the switch can synchronize to the most current VLAN database.

A switch that is reset or newly joined to the VTP domain will also send out an advertisement request.

(Reference: http://www.cisco.com/c/en/us/support/docs/lan-switching/vtp/10558-21.html#vtp_msg)

* * *

Configuring VTP

By default, a switch is in VTP *server* mode, and joined to a *blank* domain labeled *NULL.*

To change the VTP *domain* name:

Switch(config)# *vtp domain MYDOMAIN*

Note that the domain name is case sensitive. To configure the VTP *mode*:

Switch(config)# *vtp mode server*
Switch(config)# *vtp mode client*
Switch(config)# *vtp mode transparent*

The VTP domain can be secured using a password:

Switch(config)# *vtp password P@SSWORD!*

The password is also case sensitive. All switches participating in the VTP domain must be configured with the same password. The password is hashed into a 16-byte MD5 digest.

Cisco switches use **VTP version 1** by default, which is not compatible with VTPv2**.** The VTP version is dictated by the **VTP server**, and if the server is configured for VTPv2, all other switches in the VTP domain will change to v2 as well.

Switch(config)# *vtp version 2*

To view status information about VTP:

Switch# *show vtp status*

```
VTP Version                     : 2
Configuration Revision          : 42
Maximum VLANs supported locally : 1005
Number of existing VLANs        : 7
VTP Operating Mode              : Server
VTP Domain Name                 : MYDOMAIN
VTP Pruning Mode                : Disabled
VTP V2 Mode                     : Enabled
VTP Traps Generation            : Disabled
MD5 digest                      : 0x42 0x51 0x69 0xBA 0xBE 0xFA 0xCE 0x34
Configuration last modified by 0.0.0.0 at 6-22-14 4:07:52
```

To view VTP statistical information and error counters:

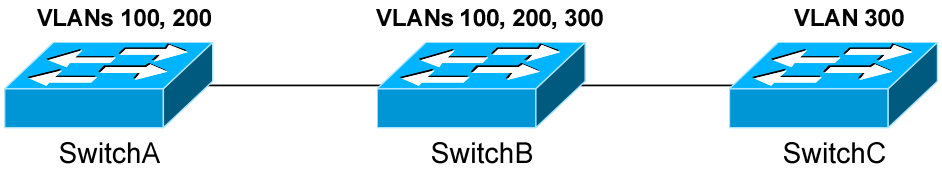
Switch# *show vtp counters*

* * *

VTP Pruning

Recall that Layer-2 switches belong to only *one* **broadcast domain.** A Layer-2 switch will thus forward both broadcasts and multicasts out *every* port in the same VLAN but the originating port. This includes sending out broadcasts out trunk ports to *other* switches, which will in turn flood that broadcast out all ports in the same VLAN.

VTP pruning eliminates unnecessary broadcast or multicast traffic throughout the switching infrastructure. Consider the following example:



Assume that a host is connected to SwitchB, in VLAN 300. If the host sends out a broadcast, SwitchB will forward the broadcast out every port in VLAN 300, including the trunk ports to SwitchA and SwitchC. Both SwitchA and SwitchC will then forward that broadcast out every port in VLAN 300.

However, SwitchA does not have any ports in VLAN 300, and will drop the broadcast. Thus, sending the broadcast to SwitchA is a waste of bandwidth.

VTP pruning allows a switch to learn which VLANs are *active* on its neighbors. Thus, broadcasts are only sent out the necessary trunk ports where those VLANs exist. In the preceding example, pruning would prevent VLAN 300 broadcasts from being sent to SwitchA, and would prevent VLAN 100 and 200 broadcasts from being sent to SwitchC.

VTP pruning is **disabled by default** on IOS switches. VTP pruning must be enabled on a server, and will be applied globally to the entire VTP domain:

Switch(config)# *vtp pruning*

Both VLAN 1 and the system VLANs 1002-1005 are never eligible for pruning. To manually specify which VLANs are pruning eligible on a trunk:

Switch(config)# *interface gi2/24*
Switch(config-if)# *switchport trunk pruning vlan 2-10*
Switch(config-if)# *switchport trunk pruning vlan add 42*
Switch(config-if)# *switchport trunk pruning vlan remove 5*
Switch(config-if)# *switchport trunk pruning vlan except 100-200*
Switch(config-if)# *switchport trunk pruning vlan none*

* * *