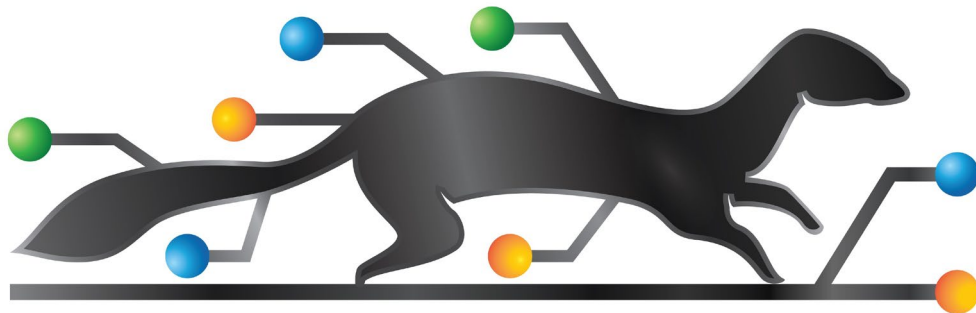


Network FerretTM

Domain Model

V12.4



Contents

Preface	i
Network Ferret License Agreement.....	i
About This Guide.....	i
Who is This Guide For?	i
Getting Additional Help.....	i
Getting More Information	i
Getting Support.....	ii
1: Objects and Relationships.....	1
Objects	1
Type.....	1
Attributes.....	1
Object Uniqueness	1
Relationships.....	2
Type.....	2
Attributes.....	3
Cardinality.....	3
Relationships-Lite.....	5
2: Abstract Data Output.....	6
Factory Shipments	6
Node	6
Type.....	6
Entities	7
Relationships	7
3: Domain Model Layouts.....	8
Visual Representation	8
Prototype Class	8
Common Attributes.....	9
4: Basic Discovery Domain Model	10
Basic Node Shipment.....	11
BGPGroup	12
BGPPeer.....	13
Board	14
Controller	15
CPU	16
DiskPartition.....	17
Drive	18
Fan	19
FileSystem.....	20
Interface.....	21
IPStack	23
ISIS	24
ISISInterface.....	25
LinkAggregation.....	26



LinkAggregator	26
LAGInterface.....	27
MaintenanceDomain.....	28
MaintenanceAssociation.....	29
MaintenanceEndPoint.....	30
Memory.....	31
Node	32
OS	34
OSPF.....	35
OSPFInterface.....	36
Port.....	37
PowerSource	38
Program.....	39
Shelf	40
Traffic Policer.....	41
Traffic Policer Interface.....	42
SNMPCredential	43
VCendpoint.....	44
VirtualIPStack	45
VirtualRouter.....	46
VLAN	47
5: MPLS Domain Model.....	48
LS (Label Space).....	49
LSInterface	50
PWVirtualCircuit.....	51
ServiceInstance	52
VRF	53
VRFInterface.....	54
VRFRouteTarget.....	55
6: SDN Domain Model.....	56
SDComponent (SDController, SDManager, SDRouter).....	57
SDInterface.....	57
SDVPN	58
SDPeer (sdControl, sdOMP, sdBFD).....	58
7: Services Domain Model.....	59
Customer	60
SAP (Service Access Point).....	61
SDP (Service Destination Point).....	62
Service.....	63
ServiceInterface.....	64
8: Advanced Discoveries	65
Types of Objects	65
Connection IDs.....	65
Connection Type	65
Type of Connections	66
Physical Connections	66



Logical Connections	66
Unknown Connections	66
What is Reported.....	66
9: Switch Connectivity	67
Remote Information.....	67
Switch Shipment.....	68
Dumb Repeater Shipment.....	69
Port – uplink.....	70
Port – host	70
Unconnected IPs Shipment.....	71
IP	71
Bridge MAC Table Shipment.....	72
Port	72
VLAN Group Shipments.....	73
Node	73
Interface.....	74
10: Router Connectivity	75
Router Shipment	75
BFDEndpoint	76
BGPPeer.....	76
ISISAdjacency	77
LDPPeer	78
LSP	79
OSPFNeighbor	80
Interface.....	80
PWVirtualCircuit.....	81
Tunnel.....	82
VCEndpoint.....	82
11: General Physical Connectivity	83
Connections Shipment	83
Interface.....	83
MaintenanceEndPoint.....	84
FEX Shipment	85
Board	85
12: Interface Polling Domain Model.....	86
Device Shipment	87
Interface.....	87
13: VMWare Domain Model	88
DataCenter Shipment.....	89
VMWareDataCenter	90
VMWareHostSystem	90
VMWareDataStore.....	91
VirtualMachine	92
VMWareNetworkStorage	93
14: ATM Domain Model.....	94
ATM Shipment.....	94



Interface.....	94
15: Handling Non-standard Devices.....	95
SNMP Standards.....	95
Common Model.....	95
VSP Files.....	96
Location.....	96
Format.....	96
Inheritance.....	96
Customization.....	97
VSP File Parameters.....	97
16: Standard SNMP MIBs and Vendor Support.....	98
Basic MIB II 1.3.6.1.2.1.....	98
IF-MIB MIB2.2.....	98
AT-MIB MIB2.3.....	98
IP-MIB MIB2.4.....	98
Transmission-MIB MIB2.10.....	99
MPLS-MIB MIB2.10.166.....	99
OSPF-MIB MIB2.14.....	99
OSPFv3-MIB 1.3.6.1.3.102.....	99
BGP-MIB MIB2.15.....	100
Bridge-MIB MIB2.17.....	100
Repeater-MIB MIB2.22.....	100
Host Resources-MIB MIB2.25.....	100
EXT-IF-MIB MIB2.31.....	101
ATM-MIB MIB2.37.....	101
Entity-MIB MIB2.47.....	101
IPv6-MIB MIB2.55.....	102
VRRP-MIB MIB2.68.....	102
IGMP-MIB MIB2.85.....	102
ISIS-MIB MIB2.138.....	102
BFD-MIB no standard.....	102
CFM-MIB 1.3.111.2.802.1.1.8.1.....	102
LLDP-MIB 1.0.8802.1.1.2.1.....	103
LAG-MIB 1.2.840.10006.300.43.....	103
Vendor Not related to any standard.....	103



Preface

This preface contains background information that you should know before using this Guide.

Network Ferret License Agreement

Read The Terms And Conditions Of The License Agreement Found In The Network Ferret User Guide.

About This Guide

The chapters of this guide contain explanations, reference information, and examples of the Network Ferret domain model. The domain model represents the data that Network Ferret generates. The domain model includes generic information valid across all devices as well as vendor-specific information.

Who is This Guide For?

This Guide is intended for professional development teams (programmers, technical writers and quality assurance engineers) who want to embed Network Ferret in their own management applications.

This Guide assumes that readers already understand:

- The basics of SNMP and Network Management.
- The basic of networking including the definitions of layer 2, layer 2, etc.
- The basics of computer systems including concepts such as file systems, disk drives, etc.
- The basics of Network Ferret as described in the Network Ferret User Guide.

This Guide assumes readers already have:

- Access to a copy of the Network Ferret installation media.

Getting Additional Help

The following sources of additional assistance and information are available to all Network Ferret users.

Getting More Information

You can find additional information about Network Ferret in the following related publications:



- *Network Ferret User Guide*: Intended for all users who want to understand what Network Ferret is and how to use it.
- *Network Ferret Programmer Guide*: Intended for programmers who want to extend the functionality of Network Ferret using custom Java code.

Getting Support

When contacting support, be prepared with the necessary information that will make handling your problem easier. You should have:

- `Discovery.txt` – this may be called something else by your application. This file can get rather large. If shipping the file to support, it is best to zip it up. If the problem is regarding a specific device, support may have you sort the log and pull out the information for that device rather than sending the entire log.
- `Exceptions.txt` – this may be called something else by your application.
- In some cases support may ask you to zip up the config, log and report directories.



1: Objects and Relationships

This chapter describes the fundamental concepts that will be used to describe the Network Ferret domain model.

Objects

Type

An object is exactly what you would think it is. An object represents a thing that is discovered by Network Ferret. Each object has a type. Examples of types include Node, Board and Interface.

An object will sometimes be referred to as entity or an instance.

An object type will sometimes be referred to as a class.

A subclass denotes a specialization of a more general class. For example, a square is a subclass of rectangle. There are no subclasses in Network Ferret. You may choose to create subclasses in your own application but that is your decision to make.

Attributes

Objects have attributes. Attributes are descriptive information about an object. Examples of attributes are: snmpIndex, speed and capacity.

In theory, all objects of the same type will have the same attributes. In practice, objects of the same type in Network Ferret will often not have all of the attributes that they could have. The reason for this is that some attributes can only be found in private vendor MIBs and not all devices in Network Ferret will have vendor-specific support.

Object Uniqueness

The ability must exist to uniquely identify an object. In the world of networking this is very difficult. There is no one attribute that can



uniquely identify an object. There are several attributes which people commonly think are unique but are not.

IP Addresses are not always unique. Two networks using the same private network number will have IP Addresses in common. A less relevant example is the case where switches all have the same IP Address assigned to their back up management port.

MAC Addresses are not always unique. The Token Ring protocol software assigns MAC addresses to NICs performing specific functions on the ring. These same MAC addresses will exist on multiple rings. The Solaris operating system software assigns the same MAC Address to all NICs in a machine.

SNMP sysNames are not always unique because not all machines have a sysName and the ones that do are assigned by humans who make mistakes.

SNMP indexes are unique within a machine but they are also subject to change should the machine be rebooted. So they are unique at any given point in time but the object with index 5 now is not guaranteed to be the same object that had index 5 yesterday.

The user of Network Ferret has more of a challenge than Network Ferret itself. Network Ferret reports what it finds at the time it runs so it has no concern or knowledge of changing SNMP indexes. By the rules of IP, Network Ferret cannot interrogate two networks with the same network address at the same time so Network Ferret has no major concern for duplicate IP Addresses. Users of Network Ferret must come up with some unique naming strategy and a strategy for reconciling SNMP indexes from one discovery to the next. The Network Ferret Topology feature, which is separate from the discovery engine, implements algorithms to determine differences between discoveries.

Relationships

Type

A relationship describes a linkage between two and only two objects. These two objects are called the parent and the child. Relationships have a type which is used to describe the nature of the linkage between the parent and child. Relationship type names tend to be verb-like actions. For example, an Outlet powers a Lamp where powers is the relationship. The parent, the Outlet, is the one providing the linkage to the child, the Lamp.



Relationships tend to fall into two broad categories. Grouping relationships define collections of objects. For example, a Board collects a set of Ports. Dependency relationships define a need of one object for another. For example, a physical Disk Partition hosts a logical File System.

Attributes

Instances of relationship types have only two attributes. They have a reference to a parent object and a reference to a child object.

In theory, a relationship can have other attributes but this is not implemented in Network Ferret. This does not mean that the embedding application cannot implement this functionality in its own data model.

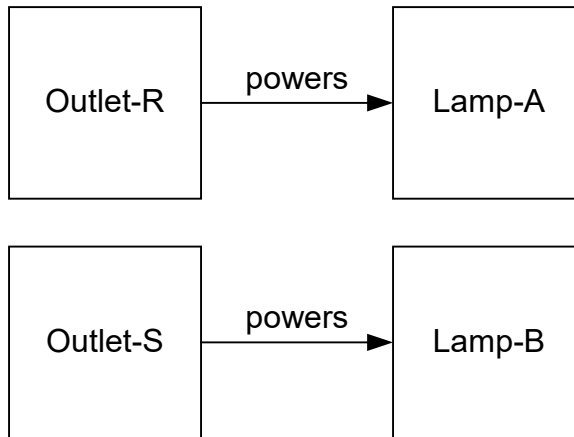
Cardinality

Relationships have cardinality. Cardinality defines the number of objects that can participate as a parent or as a child. Suppose:

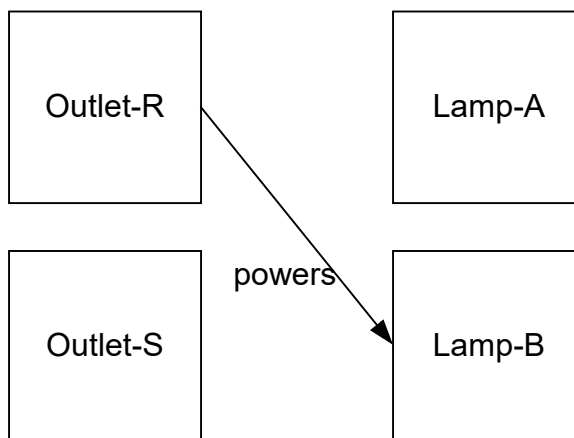
- A Lamp class with two instances of Lamp called lampA and lampB.
- An Outlet class with two instances of Outlet called outletR and outletS.
- The powers relationship type.
- Outlet participates as a parent.
- Lamp participates as a child.
- An Outlet can only power one thing at a time.
- A Lamp can only be powered by one thing at a time.

We can now say that lampA is plugged into outletR. In other words, outletR powers lampA. We can also say that OutletS powers lampB.





If we now say that outletR powers lampB, this implies that outletR no longer powers lampA and outletS no longer powers lampB since an Outlet can only power one thing at a time and a Lamp can only be powered by one thing at a time.



This is known as a one-to-one cardinality.

There are three other flavors of cardinality:

One-to-many: A parent can have many children but a child can have only one parent. For example, a Subnet can collect many IP addresses but an IP address can be in only one Subnet.

Many-to-many: A parent can have many children and a child can have many parents. For example, a VLAN can contain many switch ports and a switch port can be a member of many VLANs.

Many-to-one: A parent can have only one child but a child can have many parents. There are no examples of this cardinality in Network Ferret.



Cardinality provides a strong hint as to how a relationship might be implemented in a relational database.

Relationships-Lite

In the Network Ferret advanced discoveries which concern themselves with connections between devices, we found that it was much easier to dispense with the relationships and simply create two objects each of which has attributes that refer to the other. For example, `remoteDiscoveryIP` and `remoteIndex`.



2: Abstract Data Output

This chapter defines the output of Network Ferret in an abstract manner. No reference is made to any particular protocol (such as HTTP) or data format (such as XML).

Each chapter describing the domain model will also describe how the data is outputted.

Factory Shipments

Network Ferret outputs objects called FactoryShipments. A Factory Shipment (shipment) is logically a set of key/value pairs we will call a hashtable. All of the keys and values will be strings.

The Programmer's Guide describes the API used to access the data in a FactoryShipment.

A shipment usually contains information about a single node. The exception is some of the advanced discoveries whose specific output will be discussed below.

All of the objects within the shipment will have unique identities within their own class with respect to this node. For example, all Interfaces will have unique SNMP indexes within a Node. All Ports will have a unique SNMP index within a Node. It is very likely that some Interfaces will have the same SNMP index as a Port. These indexes are NOT unique across nodes. Just about every Node will have an Interface with an SNMP index of 1.

A Factory shipment contains the following keys and values:

Node

Node is the name for the shipment. For basic discovery, the name will be the discovery IP. This is a String. For historical reasons, but necessarily good ones, it is called node rather than name.

Type

Type defines the nature of the shipment. More specifically, type defines which part of Network Ferret produced the shipment. See AtiFactoryNames.java in the java/src directory for the Java constants.



- Basic discovery
- Layer 3 connectivity discovery
 - Layer 3 connections
 - Subnet groupings
 - Layer 3 LAN connections
- Layer 2 connectivity discovery
 - Layer 2 connections
 - Dumb Repeaters
 - Unconnected IPs
 - VLAN Groups
 - MACs
- VMWare discovery
 - VMWare Datacenter
- ATM connectivity discovery

The type is a String. The AtiFactoryNames class shipped with Network Ferret defines constants for each of the shipment types.

Entities

Each shipment may contain a set of entities or objects. Entities of the same type or class will be grouped by the same key. That key will have the form **ENT_type** where **type** is a type defined in the domain model. Some examples of type are: ENT_NODE, ENT_BOARD and ENT_INTERFACE.

The value of the key is a collection of instances of that type. Each instance is a hashtable containing the keys and values defined in the domain model.

Relationships

Each shipment may contain a set of relationships. Relationships of the same type will be grouped by the same key. That key will have the form **REL_type** where **type** is a type defined in the domain model. Some examples of type are: REL_CONTAINS, REL_ACCESS and REL_CONTROLS.

The value of the key is a collection of instances of that relationship type. An instance of a relationship is a hashtable that defines four attributes which relate two entities. The four attributes consist of the type and unique identifier for each entity. It is not possible to have additional attributes on relationships.



3: Domain Model Layouts

This chapter describes the tables that will be contained in the following chapters.

Visual Representation

Some chapters will begin with a diagram that is a visual representation of all objects and relationships for that chapter. The boxes represent object types with their listed attributes. The lines represent relationships between the objects. A relationship line will have a one (1) or an asterix (*) on either end. This represents a one-one, one-many or many-many relationship.

Prototype Class

Description: A description of the class and any technical notes about the class. This section may refer back to the [Examples And Discussion](#) section.

Existence: When will this object appear. Some objects are only created if there is vendor-specific support for that device. Some objects only exist in network gear. Some objects only exist in non-network gear.

Unique Identifier: Every instance will have a name which, along with the Class, uniquely identifies this instance within a Node. Typically, this will be an SNMP index. This name is not an attribute within the entity. This name only serves as a unique identifier for the EntityOrder within the shipment. This is different from the ATTR_NAME attribute which only a few object types have.

Attributes: This will be a table with a row for each attribute. The columns will be:

Name – the name of the attribute as Network Ferret reports it.

Units – all attributes are reported as Strings. This defines how to interpret the String. Possible values are: text, boolean, number, integer, IP, MAC, OID, megabytes, megahertz or some other such common computer capacity measurement.



Enumeration – if this attribute has a fixed enumeration of possible values, they will be listed here.

Description – a description of the attribute.

Relationships: This will be a table with a row for each relationship type the Class participates in. If the Class participates in both directions of a relationship type, it will be listed twice. The columns will be:

Name – the name of the relationship type.

Direction – the possible values are parent and child.

Cardinality – the cardinality of the relationship type. Possible values are 1-1, 1-m, m-n.

Participants – the other Classes that can participate with this Class in this relationship.

Description – the description of the relationship type.

Common Attributes

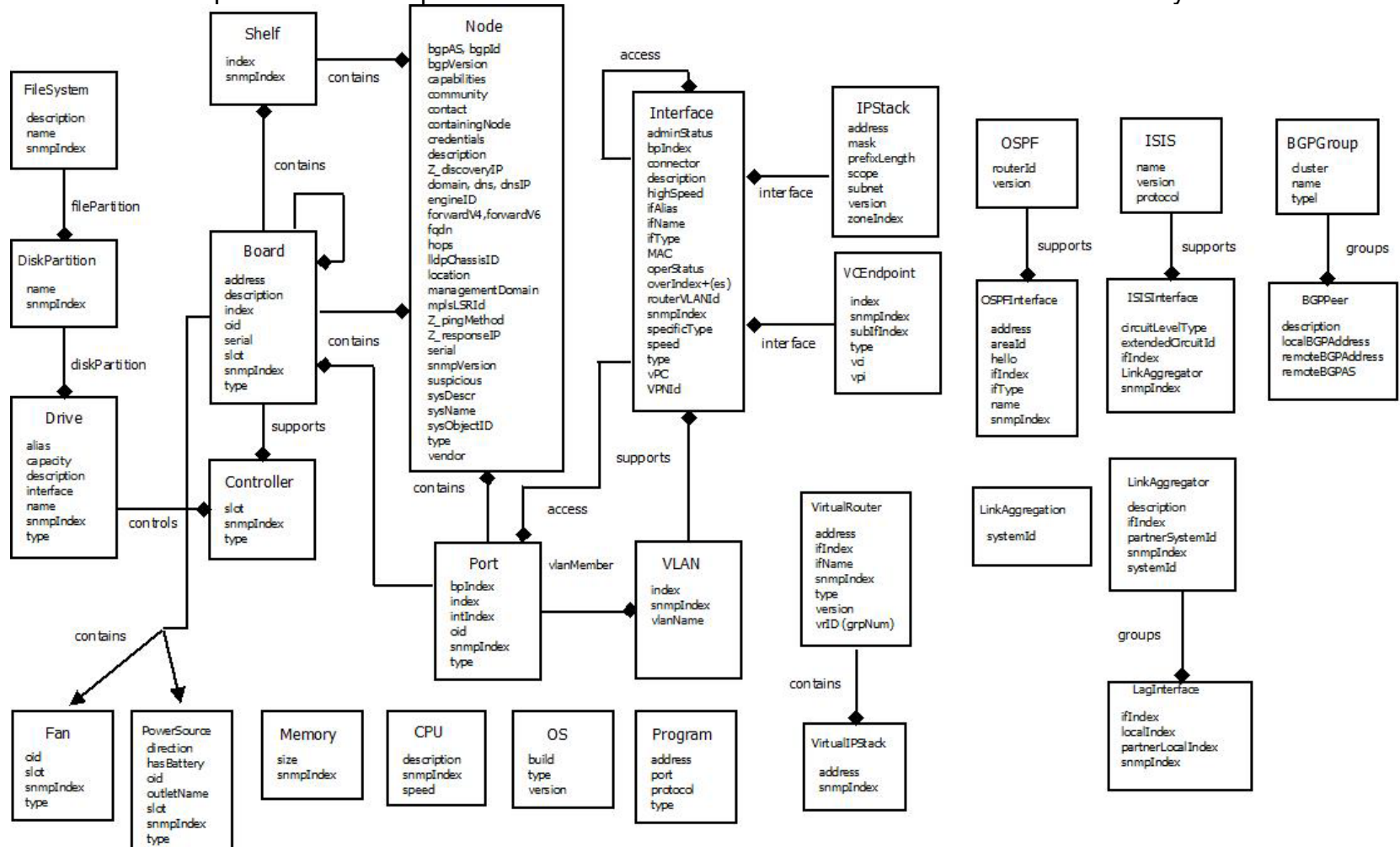
The following attributes appear in many classes and will have the same definition.

Name	Units	Enum	Description
index	OID	-	A secondary SNMP index for the object. Programmers write MIBs and they are human and they don't always talk to each other. So one object may have two indexes. One for one table and another for another table. For example, Cisco VLANs are sometimes referred to by their VLAN index (5) and sometimes they are referred to by their ManagementDomain and VLAN index (1.5). Cisco Mngt Domain are always 1.
snmpIndex	OID	-	The SNMP index used to discover this object.
type	text	-	This is not Class. This is a vendor-specific String which defines a model or a product name. If Network Ferret has vendor-specific code for a particular node and the type is not one that Network Ferret recognizes, the value will be "unknown".



4: Basic Discovery Domain Model

This chapter defines the specifics of the Network Ferret domain model for Basic Discovery.



Basic Node Shipment

Basic discovery produces a single shipment type where the type is SHIPMENT_BASIC (“**basicNode**”) and the node is the discovery IP. Each shipment will represent a single physical device.

There are a number of features that can be enabled/disabled in basic discovery. Below is a list of some features and the objects they impact. See defaults.cfg for all features that can be enabled/disabled.

Feature	Feature Name	Object Types
ext-hr	Host Resources	Controller, CPU, Drive, DiskPartition, FileSystem, Memory. Note that these objects are also created in standard general vendor processing.
ext-rtpr	Router Protocols	OSPF, OSPFInterface, bgp attributes in Node
ext-mpls	MPLS	LS, LSInterface, VRF, VRFInterface, VRFRouteTarget, mplsLSRId in Node
ext-port	Port discovery	Program. A few Programs are also created by some other features.
ext-vc	Virtual Curcuits	VCEndpoint
ext-vlan	VLANs	VLAN, managementDomain in Node



BGPGroup

Description: This type represents a BGP Peer Group.

Existence: Only for Juniper with NetConf.

Unique Identifier: name

Attributes

Name	Units	Enum	Description
cluster	text		The cluster this group belongs to. May not be present.
name	text		The name of the group.
type	text		The value of the type field or "unknown"

Relationships

Name	Dir	Card	Participants	Description
groups	parent	1-m	BGPPeer	



BGPPeer

Description: This type represents a BGP Peer that is a member of a group. This is not to be confused with the BGPPeer object coming out of L3 discovery which is actually reporting connectivity. The BGPPeer object here uses a subset of the attributes.

Existence: Only for Juniper with NetConf.

Unique Identifier: remoteBGPAddress

Attributes

Name	Units	Enum	Description
description	text		
localBGPAddress	IP		
remoteBGPAddress	IP		
remoteBGPAS	text		

Relationships

Name	Dir	Card	Participants	Description
groups	child	1-m	BGPGroup	



Board

Description: This type represents a physical board in a device. Board may also be referred to as Card in some places. A Board is generally removable from a Node whereas Shelves are generally fixed in a Node and Ports are generally fixed in a Board. A Board's sole purpose is to provide containment for a Port.

Existence: See [Containment](#) in the Examples and Discussion section. Present in private vendor MIBs and the Entity MIB (RFC-2037).

Unique Identifier: snmpIndex or index or a vendor-specific value

Attributes

Name	Units	Enum	Description
address	IP		An IP address inside the Node that this Board represents. See more detail in the Containment section of Examples and Discussion. This is rare and is currently only seen with Cisco RSM and MSM cards in switches.
description	text		
index	OID		See Common Attributes
oid	OID		An OID representing the type of board. Usually an OID existing in a private vendor MIB.
serial	text		In some vendor-specific cases the serial number of the board is retrieved.
slot	number		The slot number of a Board. Found in the EntityMIB and vendor-specific MIBs.
snmpIndex	OID		See Common Attributes
type	text		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
contains	both	1-m	Board, Board	See Containment in the Examples and Discussion section.
contains	child	1-m	Node , Board	See Containment in the Examples and Discussion section.
contains	parent	1-m	Board, Port	See Containment in the Examples and Discussion section.
contains	child	1-m	Shelf , Board	See Containment in the Examples and Discussion section.
supports	parent	1-1	Controller	This is the Board that the Controller software runs on.



Controller

Description: This type represents software which controls a drive or other hardware. A controller can either be part of the OS or the software can live on a physical Board.

Existence: Controllers will only exist in non-network Nodes such as servers that are running a supported system agent. Currently, this is on CompaqInsightManager.

Unique Identifier: snmpIndex

Attributes

Name	Units	Enum	Description
slot	number		
snmpIndex	OID		See Common Attributes in this section.
type	text		See Common Attributes in this section.

Relationships

Name	Dir	Card	Participants	Description
controls	parent	1-1	Drive	This relationships defines the Drive that this Controller software controls.
supports	child	1-1	Board	If this Controller software runs on a Board rather than in the OS, this relationship will exist. It defines the Board that this software runs on.



CPU

Description: This type represents a computer processor. There could be multiple CPUs in a Node.

Existence: CPUs are found in a few vendor devices. CompaqInsightManager for regular hosts and a couple of network device vendors.

Unique Identifier: snmpIndex

Attributes:

Name	Units	Enum	Description
description	text		
snmpIndex	OID		See Common Attributes
speed	MHZ		The speed of the CPU in megahertz.

Relationships

Name	Dir	Card	Participants	Description
none				



DiskPartition

Description: This type represents a physical partition of a Drive. In the Unix world there is a clear differentiation between partitions and file systems. In Windows the differentiation is not so clear. A partition in Windows would be called something like C: which is also the name of the file system.

Existence: DiskPartitions will only exist in non-network Nodes such as servers that support the HostResources MIB (RFC-2790). This class is not created by Compaq Insight Manager.

Unique Identifier: name, and if none, then snmpIndex

Attributes:

Name	Units	Enum	Description
name	text		The name of the partition
snmpIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
diskPartition	child	1-m	Drive	This relationship defines the Drive that this DiskPartition lives on.
filePartition	Parent	1-m	FileSystem	This relationship defines the FileSystems that are hosted on this DiskPartition.



Drive

Description: This type represents a storage device. Typically this is a hard disk drive, CDROM or a floppy drive.

Existence: Drives will only exist in non-network Nodes such as servers that support the HostResourcesMIB (RFC-2790), CompaqInsightManager or a few vendor-specific MIBs.

Unique Identifier: snmplIndex

Attributes:

Name	Units	Enum	Description
alias	text		Only in Sun Management Center.
capacity	GByte		The size of the drive in gigabytes. Not in Sun Management Center
description	OID		Not in Sun Management Center
interface	text	IDE, SCSI	The bus interface type for the Drive. Not in Sun Management Center
name	text		The name of the drive
snmplIndex	OID		See Common Attributes
type	text	fixed, floppy, removeable, driveArray	See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
controls	child	1-1	Controller	This relationship defines the Controller that manages this Drive.
diskPartition	parent	1-m	DiskPartition	This relationship defines the DrivePartitions that this drive is partitioned into. Note that a FileSystem cannot live on a Drive by itself. A FileSystem must be hosted by a DrivePartition.



Fan

Description: This type represents a Fan in a device.

Existence: Fans will exist in devices properly supporting the Entity MIB (RFC-2037) and a few vendor-specific MIBs.

Unique Identifier: snmpIndex

Attributes:

Name	Units	Enum	Description
oid	OID		An OID representing the type of fan. This is the vendorType variable.
slot	int		The slot of the fan
snmpIndex	OID		See Common Attributes
type	text		Either the vendorType variable or a name from a vendor-specific MIB

Relationships

Name	Dir	Card	Participants	Description
contains	Child	1-1	Board	Normally, a Fan has no relationship and it is implied to be in the Node. If this relationship exists, it will be to a Board.



FileSystem

Description: This type represents a logical file system on a storage device. In Unix examples of FileSystem are /, /usr and /var/spool. In Windows examples of FileSystem are C: and D:.

Existence: FileSystems will only exist in non-network Nodes such as servers that support the HostResourcesMIB (RFC-2790) or CompaqInsightManager.

Unique Identifier: name, and if none, then snmpIndex

Attributes:

Name	Units	Enum	Description
description	text		Compaq Insight Manager only
name	text		The name of the file system
snmpIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
filePartition	Parent	1-m	FileSystem	This relationship defines the FileSystems that are hosted on this DiskPartition.



Interface

Description: This type represents a layer 2 interface on a device. See more detail in the [Layer 2](#) section of Examples and Discussion. Interfaces in SNMP Nodes will have more attribution than non-SNMP Nodes.

Existence: Interfaces will exist in every Node whether it supports SNMP or not. The only exception is non-managed hubs. See more detail in the [Internal Node Scenarios](#) section of Examples and Discussion.

Unique Identifier: snmpIndex

Attributes:

Name	MIB	Units	Enum	Description
adminStatus	MIB-II	int	1, 2, 3	1=up, 2=down, 3=testing
bpIndex	MIB-II	int		The bridge port index. Only interfaces in switches will have this attribute.
connector	RFC-2863	boolean		Does this interface have a physical connector. Not always accurate
description	MIB-II	text		
highSpeed	RFC-2863	MB/S		Should be used for high-speed interfaces in place of ifSpeed (speed)
ifAlias	RFC-2863	text		Non-volatile, manager-supplied name for the interface
ifName	MIB-II	text		
ifType	MIB-II	int		See the file ianaIfTypes in AMT/config folder
MAC	MIB-II	MAC		Some devices require NF to set the MACs as the vendor does not bother populating this value in the MIB. This usually happens with switches.
operStatus	MIB-II	int	1, 2, 3	1=up, 2=down, 3=testing
overIndex	MIB-II	int		The SNMP index of the interface on which this interface runs, if any. Subinterfaces are also reflected in the relationships in the shipment so it is not important to pay attention to this value.
overIndexes	MIB-II	List of int		Comma separated list of parent ifIndexes. If this is present, overIndex should be ignored as it will contain only one of the indexes in this list. Multiple parents generally occurs in LAG situations.
routedVLANId		int		The ID of the VLAN that this interface routes. The VLAN should be defined in the switch that this interface is attached to. Only extracted from Cisco routers.
snmpIndex	MIB-II	int		See Common Attributes
specificType		text		This is different from type in every other Class. This is the actual string defined by the IANA ifType MIB. See the file ianaiftypes in the AMT/config folder.
speed	MIB-II	int		The value of the ifSpeed variable. There are cases where this is set by a human so it is not guaranteed to be accurate.
type		text		This is different from type in every other Class. This is a textual translation of ifType. The name can be found in the ianaiftypes file in the AMT/config folder.
vPC		text		The value is specific to CiscoNX. The existence of this attribute means that the interface can have a mirror on another device.



VPNId		text		The value is specific to Viptella using the REST API.
-------	--	------	--	---

Relationships

Name	Dir	Card	Participants	Description
access	parent	1-m	Interface	This relationship defines subinterfaces that sit on top of this interface. See more detail in the Internal Node Scenarios of Examples and Discussion.
access	child	1-m	Interface, Port	This relationship defines two scenarios. In the first scenario, the Interface is sitting on top of a Port. In the second scenario, this Interface is a subinterface sitting on top of another Interface. See more detail in the Internal Node Scenarios of Examples and Discussion.
interface	parent	1-m	IPStack , VCEndpoint	This relationship defines the IPs and VCEndpoints that sit on top of this Interface. See more detail in the Internal Node Scenarios of Examples and Discussion.
supports	parent	m-m	VLAN	This relationship defines the VLANs that use this Interface for trunking. See more detail in the VLAN section of Examples and Discussion.



IPStack

Description: This type represents an IP stack running on a device. There can be many IPStacks in one Node and many stacks associated with a given interface.

Existence: At least one IPStack will exist in every Node whether it supports SNMP or not. The only exception is non-managed hubs. See more detail in the [Internal Node Scenarios](#) section of Examples and Discussion.

Unique Identifier: address.

Attributes:

Name	Units	Enum	Description
address	IP		The IP Address in v4 or v6 format.
mask	IP		The subnet mask in dotted decimal notation. V4 only.
prefixLength	int		Only present in v6 objects.
scope	ENUM		V6 only. LL, SL, MC, GUA, OTHER. LinkLocal, SiteLocal, Multicast, Global Unicast Address, Other.
subnet	IP		The subnet in dotted decimal notation. V4 only.
version	int		4 or 6
zoneIndex	IP		only present in v6 objects.

Relationships

Name	Dir	Card	Participants	Description
interface	child	1-m	Interface	This relationship defines the Interface that this IPStack sits on top of.



ISIS

Description: This object represents an instance of ISIS running in a router. There can be multiple instances of this object. The unique id of these objects will be the name.

Existence: Will exist in routers running ISIS (RFC-4444). The RoutingProtocols basic extension must be running.

Unique Identifier: name.

Attributes:

Name	Units	Enum	Description
name	text		
protocols	text		Comma separated (one or more of these). iso8473, ipv4, ipv6
version	text		version.level

Relationships

Name	Dir	Card	Participants	Description
supports	parent	1-m	ISISInterface	The Interfaces used for ISIS.



ISISInterface

Description: This object represents an interface that is supporting ISIS.

Existence: Will exist in routers running ISIS (RFC-4444). The RoutingProtocols basic extension must be running.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
circuitLevelType	int		See MIB
extendedCircuitId	int		See MIB
ifIndex	int		The ifIndex of the interface supporting ISIS.
LinkAggregator	OID		Some ISIS implementations will stack the ISIS interface over each interface participating in LinkAggregation. Rather than reporting all interfaces, we report the OID of the LinkAggregator instead.
remoteMAC	MAC		Populated in certain situations. Used to match ISIS adjacencies.
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
supports	child	1-m	ISIS	The Interfaces used for ISIS.



LinkAggregation

Description: This object represents the existence of LinkAggregation within a device.

Existence: Will exist in devices running LAG (IEEE8023-LAG-MIB). The LAG basic extension must be running.

Unique Identifier: name.

Attributes:

Name	Units	Enum	Description
name	text		The systemId for LAG

LinkAggregator

Description: This object represents an instance of an aggregated link within a device.

Existence: Will exist in devices running LAG (IEEE8023-LAG-MIB). The LAG basic extension must be running.

Unique Identifier: snmpIndex.

Attributes:

Name	Units	Enum	Description
description	text		
ifIndex	int		The ifIndex of the interface representing this Aggregator.
partnerSystemId	text		The partner's LAG system ID. The remote end.
snmpIndex	OID		See Common Attributes
systemId	text		The LAG system ID

Relationships

Name	Dir	Card	Participants	Description
groups	parent	1-m	LAGInterface	The Interfaces used for this aggregated link.



LAGInterface

Description: An interface participating in an aggregated link. Usually there is more than one but we have seen cases where a single interface makes up the LAG.

Existence: Will exist in devices running LAG (IEEE8023-LAG-MIB). The LAG basic extension must be running.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
ifIndex	int		The interface index associated with this LAG interface
localIndex	int		Used within the LAG MIB to match endpoints
partnerLocalIndex	int		Used within the LAG MIB to match endpoints
snmplIndex	int		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
groups	child	1-m	LinkAggregation	The aggregator we are grouped by.



MaintenanceDomain

Description: Part of the Connectivity Fault Management model (CFM). A maintenance domain is a sphere of influence.

Existence: Will exist in devices running CFM (IEEE8021-CFM-MIB). The CFM basic extension must be running. Generic code only tested with Cisco.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
index	int		Same as snmplIndex
level	int		0-7. The larger the domain, the higher the value
name	text		The name
snmplIndex	int		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
groups	parent	1-m	MaintenanceAssociation	MAs are part of an MD.



MaintenanceAssociation

Description: Part of the Connectivity Fault Management model (CFM). A set of MaintenanceEndPoints.

Existence: Will exist in devices running CFM (IEEE8021-CFM-MIB). The CFM basic extension must be running. Generic code only tested with Cisco.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
index	int		The index
name	text		The name
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
groups	child	1-m	MaintenanceDomain	The MA is part of an MD
groups	parent	1-m	MaintenanceEndPoint	The MA groups MEPs



MaintenanceEndPoint

Description: Part of the Connectivity Fault Management model (CFM). A maintenance endPoint is an interface that participates in CFM.

Existence: Will exist in devices running CFM (IEEE8021-CFM-MIB). The CFM basic extension must be running. Generic code only tested with Cisco.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
direction	text	up/down	
ifIndex	int		The interface index of the participating interface
index	int		The index
snmplIndex	OID		The snmplIndex

Relationships

Name	Dir	Card	Participants	Description
groups	child	1-m	MaintenanceAssociation	MEPs are part of an MA.

Memory

Description: This type represents the RAM in a device.

Existence: Memory will exist in servers that support the HostResourcesMIB (RFC-2790) or CompaqInsightManager and a few networking devices. There will only be one instance if it exists.

Unique Identifier: snmpIndex (always 0)

Attributes:

Name	Units	Enum	Description
size	MB		The amount of memory in megabytes.
snmpIndex	int	0	Always 0

Relationships

Name	Dir	Card	Participants	Description
none				

Node

Description: This type represents a physical device. See more detail in the [Containment](#) section of Examples and Details.

Existence: Network Ferret reports based on Nodes so there will always be a Node present.

Unique Identifier: None. There will only be one Node in a shipment and its name will always be “node”.

Attributes:

Name	Populated by	Units	Enum	Description
bgpAS	ext-rtpr	int		The BGP autonomous system number. Only in routers
bgpld	ext-rtpr	IP		The BGP ID of the router.
bgpVersion	ext-rtpr	csv		A comma separated list of supported versions.
capabilities	NF	csv		A comma separated list of capabilities such as router, bridge, snmp, etc. These values are populated from many features.
contact	MIB2	text		The value of sysContact
containingNode	NF	text		The node this node is contained in. There are only two cases of this. A Cisco router module running its own SNMP agent. This will be the address of the containing switch. For a Cabletron 6k card, this will be the serial number of the containing switch. There is a third case of this which has the opposite meaning. For a Dell IDRAC this attribute is the FQDN of the Node in the Dell. See readme for 10.12.
description	MIB2	text		By default, the value of sysDescr but can be overridden by a description in the Entity MIB.
Z_discoveryIP	NF	IP		The IP address on which this device was discovered
domain	dns	text		The domain portion of the DNS name of a node.
dns	dns	text		The host name portion of the DNS name of a node
dnsIP	dns	IP		The address corresponding to the DNS entry. Usually this is the same as the discovery IP.
engineID	NF	text		The SNMPv3 engine ID.
forwardV4	MIB2	boolean		Does this device do IPv4 routing. Same meaning as the router capability. The value of this attribute is not important. Its presence is what matters.
forwardV6	MIB2	boolean		Does this device do IPv6 forwarding. This attribute is only reporting the presence of this value in the device. No checking is done to see if the device is actually forwarding IPv6 packets.
fqdn	NF	text		The FQDN of the host as passed in by the embedding application in the config
hops	NF	int		The number of hops from a user supplied subnet, host, seed router.
lldpChassisId	LLDP	Text		The LLDP chassis ID for the node
location	MIB2	text		The value of sysLocation



management Domain	ext-vlan	text		Cisco only. The VLAN management domain for the switch.
mplsLSRId	ext-mpls	IP		An IP address in the router. Not always present.
Z_pingMethod	NF	text		How a Host was created in ping. See FactoryNames PING_TYPE_* for possible values.
Z_responseIP	NF	IP		Only present when a device responds on a different IP than the IP the request was sent to.
serial	various	text		A serial number for the node. Populated by the Entity MIB and various vendors.
suspicious	NF	boolean		The value is not important. The existence of the attribute is what is important. If it exists, it signifies that the discovery IP for this node did not exist in the Node's IP address table.
sysDescr	MIB2	text		The value of sysDescr
sysName	MIB2	text		The value of sysName
sysObjectID	MIB2	OID		The SNMP object id of the device minus the standard prefix. The OID will begin with the vendor's assigned number.
type	NF	text		See Common Attributes in this section.
vendor	NF	text		The value of the vendor attribute in the VSP file for this device.

Relationships

Name	Dir	Card	Participants	Description
contains	parent	1-m	Board , Node, Port , Shelf	See Containment in the Examples and Discussion section.
contains	child	1-m	Node	See Containment in the Examples and Discussion section.



OS

Description: This type represents the operating system within the device. There is only one OS in a Node.

Existence: Every Node will have an OS. Some will just be placeholders while some will have attributes that were discovered such as a version number.

Unique Identifier: 1.

Attributes:

Name	Units	Enum	Description
build	text		The build string for the OS. Set by some vendor code.
type	text		The type of OS such as IOS or Windows.
version	text		The version for the OS. There are no standards for this so it is only populated by vendor specific code. In some cases the vendor has an actual MIB variable but in many cases a best attempt is made to parse it out of the sysDescr.

Relationships

Name	Dir	Card	Participants	Description
none				



OSPF

Description: This object represents OSPF running in a router. There are two different versions of OSPF each with their own MIB. It is assumed that both versions can be operational in the same router. There can be 0, 1 or 2 instances of this object. The unique id of these objects will be the version number.

Existence: Will exist in routers running OSPF (RFCs 1253 and 2740). The RoutingProtocols basic extension must be running.

Unique Identifier: version.

Attributes:

Name	Units	Enum	Description
routerId	IP		An address within the router.
version	int	2,3	There are two versions of OSPF.

Relationships

Name	Dir	Card	Participants	Description
supports	parent	1-m	OSPFIInterface	The Interfaces used for OSPF.



OSPFInterface

Description: This object represents an interface that is supporting OSPF. Note that the standard MIB will report either an IP address or an ifIndex. If an address is reported, Network Ferret will look up the ifIndex and populate it. So an snmpIndex will either be a.b.c.d.0 or 0.0.0.0.#.

Existence: Will exist in routers running OSPF (RFCs 1253 and 2740). The RoutingProtocols basic extension must be running.

Unique Identifier: snmpIndex (address.ifIndex).

Attributes:

Name	Units	Enum	Description
address	IP		The IP address on the interface. Could be 0.0.0.0. This has nothing to do with the backbone. It means the interface is unnumbered.
areald	IP		The OSPF area ID this interface supports. 0.0.0.0 is the backbone.
hello	int		The hello interval
ifIndex	int		The ifIndex of the interface.
ifType	text	yes ->	broadcast, nbma, pointToPoint, unknown:x
name	text		address.ifIndex. This is different from snmpIndex because Network Ferret looks up the ifIndex supporting an address whereas the snmpIndex will have an ifIndex of 0 for an address.
snmpIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
supports	child	1-m	OSPF	The Interfaces used for OSPF.



Port

Description: This type represents a physical port in a device. See more detail in the [Containment](#) section of Examples and Details. Ports are like Boards in that they are discovered both in the Entity MIB (RFC-2037) and in vendor-specific MIBs.

Existence: See [Containment](#) in the Examples and Discussion section.

Unique Identifier: snmplIndex, and if none, then some combination involving the ifIndex

Attributes:

Name	Units	Enum	Description
bpIndex	int		The bridge port index for this port. Only ports in switches will have this attribute.
index	OID		See Common Attributes in this section.
intIndex	int		The index of the interface that sits directly on top of this port
oid	OID		An OID representing the type of port. Usually an OID existing in a private vendor MIB.
snmplIndex	OID		See Common Attributes
type	text		Some vendor-specific MIBs provide a port type string. This is desc from the Entity MIB.

Relationships

Name	Dir	Card	Participants	Description
access	parent	1-m	Interface	This relationship defines the Interface that sits on top of this Port. See more detail in the Internal Node Scenarios of Examples and Discussion.
connects	both	1-1	Port	This represents an internal connection between Ports in the same box.
contains	child	1-m	Board , Node	See Containment in the Examples and Discussion section.
vlanMember	child	m-m	VLAN	This relationship defines the VLANs that this Port is a member of. See more detail in the VLAN section of Examples and Discussion. The VLAN basic extension must be running.



PowerSource

Description: This type represents a power source. A PowerSource can be either an internal source for a device or a Node that provides power to other devices such as a UPS.

Existence: Power Sources will come from the Entity MIB (RFC-2037) and some vendor MIBs.

Unique Identifier: snmpIndex, and if none, then outletName

Attributes:

Name	Units	Enum	Description
direction	text	in, out, external	The direction of the power source. For internal sources, the value will be internal. For devices that provide power to others, the value can be either in or out.
hasBattery	boolean	true / false	Notes that an output power source has a battery backup.
oid	OID		The vendorType variable from the entityMIB.
outletName	text		A textual name given to the source by its management system.
slot	int		The slot of the power source
snmpIndex	OID		See Common Attributes in this section.
type	text		See Common Attributes in this section.

Relationships

Name	Dir	Card	Participants	Description
Contains	Child	1-1	Board	Normally, a PS has no relationship and it is implied to be in the Node. If this relationship exists, it will be to a Board.



Program

Description: This type represents a logical program on a device. A Program does not necessarily imply a single running process. Program could be thought of in the same terms of the capability attribute present in the Node object except that Program might have some more information associated with it such as a TCP port number.

Most Programs are created via the port scanning capability of Network Ferret. This capability is controlled via the Port Discovery parameters which are described in the Configuring Network Ferret chapter of the User Guide.

A few Programs are created by the supported SNMP system agents. These Programs represent the agents.

Existence: Programs can exist in any Node.

Unique Identifier: port number, and if none, then the type

Attributes:

Name	Units	Enum	Description
address	IP addr		Some vendors will report the specific IP stack that a program is opened on.
port	int		The TCP or UDP port the Program was discovered on.
protocol	text	tcp, udp	The protocol used to discover the Program.
type	text		For Programs discovered using Network Ferret's port scanning, this value is taken from the configuration file. For other Programs, this value is vendor-specific.

Relationships

Name	Dir	Card	Participants	Description
none				



Shelf

Description: This type represents a physical shelf in a device. See more detail in the [Containment](#) section of Examples and Details.

Existence: See [Containment](#) in the Examples and Discussion section. Only the Cisco AS5800 creates these objects.

Unique Identifier: snmpIndex or index

Attributes:

Name	Units	Enum	Description
index	OID		See Common Attributes
snmpIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
contains	child	1-m	Node	See Containment in the Examples and Discussion section.
contains	parent	1-m	Board	See Containment in the Examples and Discussion section.



Traffic Policer

Description: This type represents QOS objects.

Existence: When QOS is enabled via IPDBasicExtension.ext-qos or in NetConf. Cisco uses SNMP. Juniper uses NetConf.

Unique Identifier: name

Attributes:

Name	Units	Enum	Description
bandwidth	number		The bandwidth limit
burst	number		The burst limit
description	text		
name	text		
shaping	boolean		Is this policer doing shaping
type	text	yes ->	See below

```
public static String RATE_MBPS = "mbps";
public static String RATE_BPS = "bps";
public static String RATE_PERCENT = "percent";
public static String RATE_CPS = "cps";
public static String RATE_PT = "perThousand";
public static String RATE_PM = "perMillion";
```

Relationships

Name	Dir	Card	Participants	Description
contains	both	1-m	TrafficPolicer	Policers can contain other Policers.
supports	child	m-m	TrafficPolicerInterface	The interfaces supporting the Policer.



Traffic Policer Interface

Description: An interface that supports a TrafficPolicer. Note that a Policer can be supported by many interfaces and a given interface can support many Policers.

Existence: When QOS is enabled via IPDBasicExtension.ext-qos or in NetConf. Currently, only Juniper creates these.

Unique Identifier: snmplIndex (not really since it is coming from NetConf!!)

Attributes:

Name	Units	Enum	Description
direction	text	yes - >	See below
ifIndex	int		The ifIndex of the interface
protocol	text		
snmplIndex	OID		See Common Attributes

```
public static String DIRECTION_INPUT = "input";
public static String DIRECTION_OUTPUT = "output";
public static String DIRECTION_BOTH = "both";
public static String DIRECTION_UNKNOWN = "unknown";
```

Relationships

Name	Dir	Card	Participants	Description
supports	parent	m-m	TrafficPolicer	The Policers that this interface supports.



SNMPCredential

Description: This type represents an SNMPCredential. It contains the same information as is contained in the credential interface.

Existence: Created when IPDReportSecurityInfoInShipment config parameter is true.

Unique Identifier: the credential string which is a combination of the various attributes.

Attributes:

Name	Units	Enum	Description
index	text		A combination of the other attributes
port	int		The UDP port number used for this credential
version	int	1,2,3	The SNMP version
V1/2 attrs			
community	text		The SNMP v1 or v2 community string
V3 attrs			
authProtocol	text	true	See defaults.cfg for possible values
authPwd	text		The authorization password
context	text		The v3 context name
privProtocol	text	true	See defaults.cfg for possible values
privPwd	text		The privacy password
userName	text		The v3 user name



VCEndpoint

Description: This type represents a virtual circuit endpoint such as a Frame Relay DLCI. This is different from a Frame Relay Interface. See more detail in the [Internal Node Scenarios](#) section of Examples and Discussion.

Existence: VCEndpoints will only exist in routers. All information is found in RFC-2115 and RFC-2515 except for some specific information in Cisco. The standard MIBs do not provide a linkage between a subinterface and the vc. It only provides a linkage to the main interface. Cisco provides the linkage to the subinterface.

Unique Identifier: snmplIndex

Attributes:

Name	Units	Enum	Description
index	OID		See Common Attributes .
snmplIndex	OID		See Common Attributes
subIfIndex	int		The ifIndex of the subInterface. Cisco only.
type	text	frame, atm	The protocol of this VCEndpoint.
vci	int		Present only for ATM Virtual Channel Links. Will not be present for Virtual Path Links.
vpi	int		Present for any ATM endpoint.

Relationships

Name	Dir	Card	Participants	Description
interface	child	1-m	Interface	This relationship defines the Interface that this VCEndpoint sits on top of. See more detail in the Internal Node Scenarios section of Examples and Discussion



VirtualIPStack

Description: This type represents an IP stack used in router redundancy scenarios. Standard VRRP (RFCs 3768 and 5798) and Cisco HSRP are the two supported protocols.

Suppose there are two routers on a subnet. One has an address of 10.10.10.1 and the other has 10.10.10.2. Most hosts can only handle a single default gateway address. So a third address is used, 10.10.10.3, and the routers mediate between each other as to which one will be the primary .3 and which will be the backup. A virtual MAC address is also used so the hosts do not have to know which router is the current primary.

If two routers are participating, each will report a VirtualIPStack object in their respective factory shipments. Both objects will share the same address.

Existence: Any router supporting the protocol.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
address	IP		The virtual IP Address in dotted decimal notation. For Cisco, this will be in the private MIB.
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
contains	child	1-m	VirtualRouter	A VR can contain one or more VIPs.



VirtualRouter

Description: This type represents a virtual router used in router redundancy scenarios. See VirtualIPStack for more explanation.

A virtual router can have one or more virtual IP stacks associated with it. With Cisco HSRP, there will be only one.

Existence: Any router supporting the protocol.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
address	IP		The real IP Address associated with this virtual router.
ifIndex	int		The ifIndex of the Interface object participating in the redundancy scenario
ifName	String		The ifName of the Interface object participating in the redundancy scenario. Only present with REST API.
snmplIndex	OID		See Common Attributes
type	String		VRRP or HSRP.
version	int		The VRRP version number. HSRP does not have this.
vrID	int		The VRRP ID or the Cisco HSRP group number.

Relationships

Name	Dir	Card	Participants	Description
contains	parent	1-m	VirtualIPStack	A VR can contain one or more VIPs.



VLAN

Description: This type represents a layer 2 virtual LAN. See more detail in the [VLAN](#) section of Examples and Discussion.

Existence: There is a standard VLAN MIB (RFC-2674) as well as vendor support. The standard came long after VLANs started appearing so most support is vendor-specific. The VLAN basic extension must be enabled.

Unique Identifier: snmpIndex

Attributes:

Name	Units	Enum	Description
description	text		Only populated by some vendors
index	OID		See Common Attributes .
snmpIndex	OID		See Common Attributes
vlanName	text		An operator assigned textual name for the VLAN.

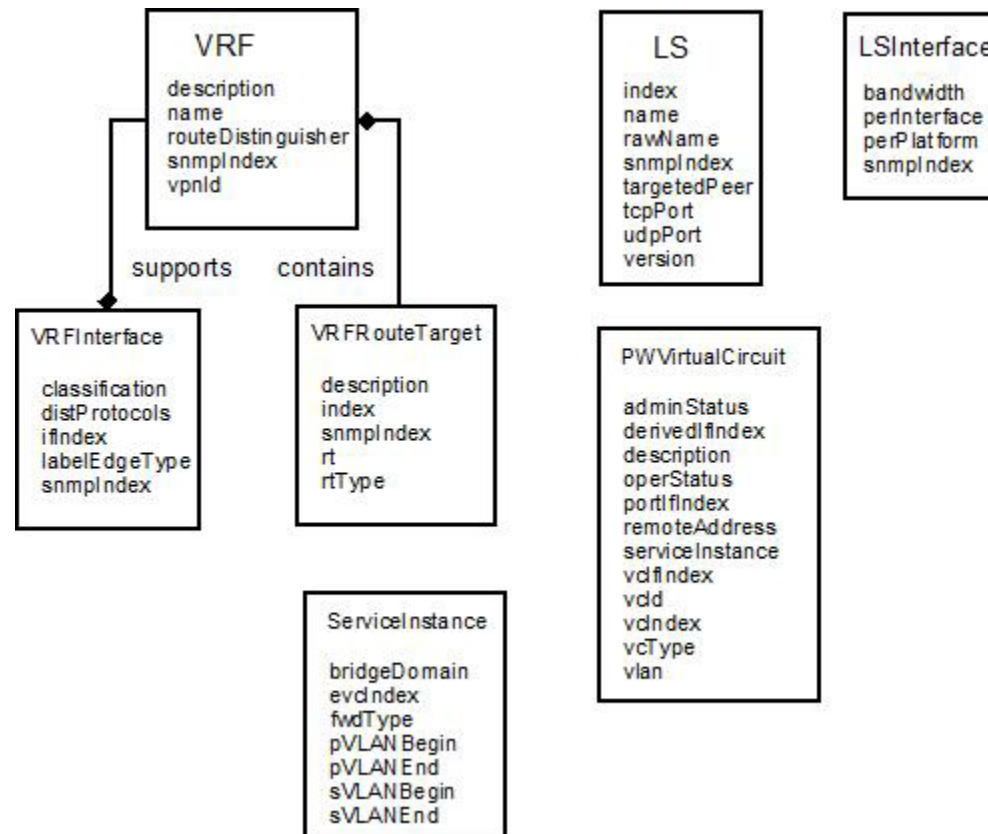
Relationships

Name	Dir	Card	Participants	Description
supports	child	m-m	Interface	This relationship defines the Interfaces that this VLAN uses for trunking.
vlanMember	parent		Port , Interface	This relationship defines the Ports or Interfaces that constitute the VLAN.



5: MPLS Domain Model

This chapter defines the MPLS objects for Basic Discovery. These objects are in the Basic Node Shipment as defined in the previous chapter. These objects were put into their own chapter because there are many of them and not all customers use the MPLS feature.



LS (Label Space)

Description: This type represents an MPLS Label Space. A Label Space has a name in the format of a.b.c.d.# where a.b.c.d is an address in the router and # is a unique id for the Label Space. An id of 0 represents the platform-wide Label Space.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled.

Unique Identifier: snmplIndex. This is the index made up of two variables: mplsLdpEntityLdpId, mplsLdpEntityIndex

Attributes:

Name	Units	Enum	Description
index	int		A unique number that makes up part of the SNMP index. Not useful on its own. mplsLdpEntityIndex
name	OID		The Label Space name – a.b.c.d.#
rawName	OID		The 6-byte OID – a.b.c.d.x.y (x and y are combined to make name) - mplsLdpEntityLdpId
snmplIndex	OID		See Common Attributes
targetedPeer	int	1, 2	1 = true, 2 = false, 0 = not present
tcpPort	int		The TCP port that is listening. A standard port #.
udpPort	int		The UDP port that is listening. A standard port #.
version	int		The version of LDP being used.

Relationships

Name	Dir	Card	Participants	Description
none				



LSInterface

Description: This type represents an Interface that is participating in MPLS. While interfaces participate with specific Label Spaces, there is no information in the MIB directly relating the two.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled.

Unique Identifier: snmplIndex. Corresponds to ifIndex.

Attributes:

Name	Units	Enum	Description
bandwidth	int		The MPLS bandwidth on the interface
perInterface	boolean	true	Does this interface participate in the per-interface LS? Only present if true.
perPlatform	boolean	true	Does this interface participate in the per-platform LS? Only present if true.
snmplIndex	int		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
none				



PWVirtualCircuit

Description: This type represents Pseudo Wire Virtual Circuits.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled. This object is reported via the L3 Shipment. Please see that section.

Unique Identifier: snmpIndex.

Attributes:

Name	Units	Enum	Description
adminStatus	int		Admin status. Same as in interface table.
derivedIfIndex	String		Either a match based on VLAN name, portIfIndex or vclIfIndex
description	String		description
operStatus	int		Operational status. Same as in interface table.
portIfIndex	int		Port ifIndex
remoteAddress	IPAddr		IP address of remote end
serviceInstance	int		See ServiceInstance
vclIfIndex	int		VC ifIndex
vcId	int		ID of virtual circuit
vcIndex	OID		SNMP index
vcType	int		Only type 5 is created. Ethernet.
vlan	int		VLAN the vc is running over

Relationships

Name	Dir	Card	Participants	Description
none				



ServiceInstance

Description: This object represents an EVC Service Instance.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled. Cisco only.

Unique Identifier: snmplIndex. Corresponds to ifIndex.

Attributes:

Name	Units	Enum	Description
bridgeDomain	uns int		The bridge domain
evcIndex	uns int		The EVC index.
fwdType	int	true	The forwarding type
ifIndex	int		The interface over which the SI runs
name	text		The SI name.
pVLANBegin	int		The beginning range of the primary VLANs
pVLANEnd	int		The ending range of the primary VLANs
snmplIndex	OID		See Common Attributes
sVLANBegin	int		The beginning range of the secondary VLANs
sVLANEnd	int		The ending range of the secondary VLANs

Relationships

Name	Dir	Card	Participants	Description
none				

VRF

Description: This type represents virtual router within a router. This works in conjunction with MPLS to provide a VPN capability. The MPLS VPN MIB is still in the draft stage so there is no RFC number. The MIB exists on the .11 branch under the main MPLS branch.

VRFs from different routers will participate in the same VPN. Essentially, the collection of VRFs is the VPN.

NOTE: There are three subclasses of VRF, VRFi and VRFrt. Currently these are only created by Juniper which has a type attribute in its private MIBS. See FactoryNames for details but in general they are VPN, L2VPN and L2VPLS.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled.

Unique Identifier: snmpIndex.

Attributes:

Name	Units	Enum	Description
description	text		
name	text		Actually embedded in the snmpIndex. The first OID is the length of the name and each subsequent OID is an ASCII character.
routeDistinguisher	text		A unique string of digits part of which is assigned by a global authority to ensure uniqueness
snmpIndex	OID		See Common Attributes
VPNid	text		The VPN that this VRF participates in.

Relationships

Name	Dir	Card	Participants	Description
contains	parent	1-m	VRFRouteTarget	The VRFRouteTargets that this VRF contains
supports	child	1-m	VRFInterface	The VRFInterfaces that participate in the VRF.



VRFInterface

Description: This type represents an Interface which supports a VRF. Since Network Ferret relationships do not have attributes, this type was created to hold the attribution. The MPLS VPN MIB is still in the draft stage so there is no RFC number. The MIB exists on the .11 branch under the main MPLS branch.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
classification	text	yes ->	carrierOfcarrier, enterprise, interProvider, unknown:x
distProtocols	csv	yes ->	A comma separated list with the following possible values: none, bgp, ospf, rip, isis, static, other.
ifIndex	int		The index of the Interface.
labelEdgeType	text	yes ->	customerEdge, providerEdge, undefined, unknown:x
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
supports	parent	1-m	VRF	The VRF that this interface supports.



VRFRouteTarget

Description: A VRF uses Route Targets to help with routing. The MPLS VPN MIB is still in the draft stage so there is no RFC number. The MIB exists on the .11 branch under the main MPLS branch.

Existence: Will exist in routers supporting MPLS (RFC-3813). The MPLS basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
description	text		
index	int		an integer that is part of the snmplIndex. Not useful on its own.
snmplIndex	OID		See Common Attributes
rt	text		The Route Target
rtType	text	yes ->	import, export, both, unknown:x

Relationships

Name	Dir	Card	Participants	Description
contains	child	1-m	VRE	The VRF that this VRFRouteTarget is in.



6: SDN Domain Model

This chapter defines the objects for the SDN Domain Model. These objects are in the Basic Node Shipment. These objects were put into their own chapter because we expect this to grow.

This initial model is based off of the Viptella MIBa. As we incorporate other vendors, this model will most likely change.

The SDN basic AND advanced discoveries need to be enabled.

There currently is no diagram because the objects do not have relationships to each other.



SDComponent (SDController, SDManager, SDRouter)

Description: The types of nodes in an SDN.

Existence: See beginning of chapter

Unique Identifier: address.

Attributes:

Name	Units	Enum	Description
address	IP		The address of the SDN node. Not necessarily the same as discovery IP
domain	text		The SDN domain ID.
site	text		The SDN site ID

SDInterface

Description: An SDN interface. Related to a real interface.

Existence: See beginning of chapter

Unique Identifier: vpnId::ifIndex.

Attributes:

Name	Units	Enum	Description
ifIndex	number		The real interface associated with this SDN interface
ifType	text		The role of this interface. Not the same meaning as a real interface ifType
protocols	text		CSV list of protocols supported by this interface such as DNS, DHCP, ICMP



SDVPN

Description: An SDN VPN

Existence: See beginning of chapter

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
name	text		Special VPN IDs will have a name such as management
snmplIndex	OID		The snmplIndex

SDPeer (sdControl, sdOMP, sdBFD)

Description: A link between two SDNComponents

Existence: See beginning of chapter. Advanced SDN discovery must run.

Unique Identifier: snmplIndex.

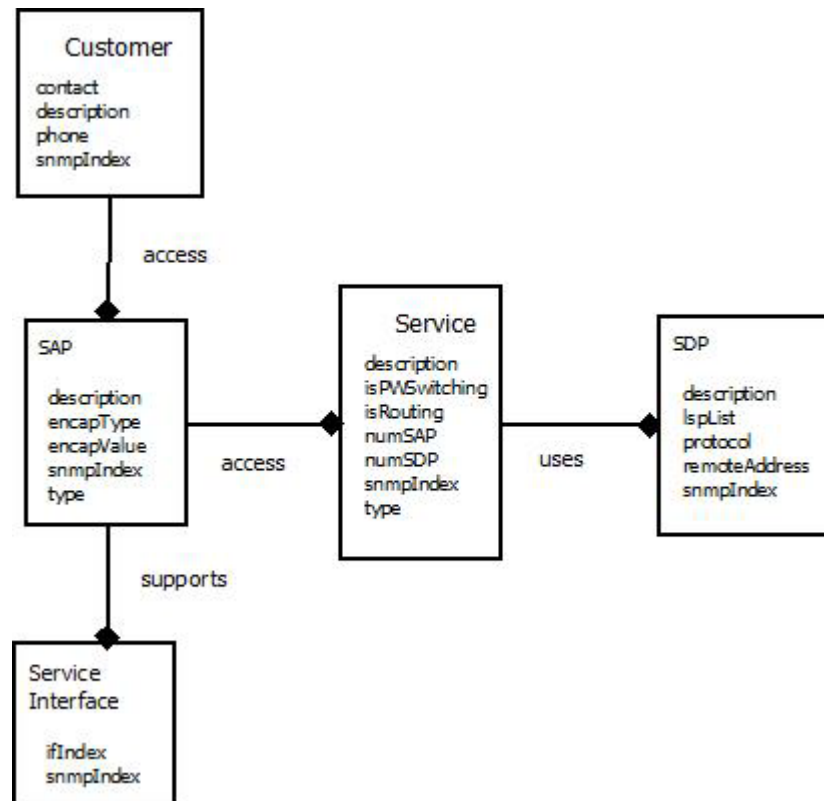
Attributes:

Name	Units	Enum	Description
remoteAddress	IP		The address of the remote SDN component
snmplIndex	OID		The snmplIndex

7: Services Domain Model

This chapter defines the objects for the Services Domain Model. These objects are in the Basic Node Shipment. These objects were put into their own chapter because there are many of them and not all customers use the Services feature.

This initial model is based off of the Alcatel/Timetra SERV MIB. As we incorporate other vendors, this model will most likely change.



Customer

Description: This type represents a customer of a service provider.

Existence: Will exist in routers supporting a services MIB. The SVC basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
contact	text		Contact name
description	text		Description
phone	text		Phone number
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
access	child	1-1	SAP	Customer has access to a service via a SAP



SAP (Service Access Point)

Description: This type represents a Service Access Point which is how a Customer accesses a Service.

Existence: Will exist in routers supporting a services MIB. The SVC basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
description	text		Description
encapType	text		Possible values include: fr, atm, mpls
encapValue	int		Vendor specific value – interpretation differs by encapType
snmplIndex	OID		See Common Attributes
type	text		Possible values include: epipe, vpls

Relationships

Name	Dir	Card	Participants	Description
access	parent	1-1	Customer	Customer has access to a service via a SAP
access	child	1-m	Service	A Customer accesses a Service via a SAP
supports	child	1-1	ServiceInterface	An SAP is associated with an interface.

SDP (Service Destination Point)

Description: There are multiple interpretations of the SDP acronym. This type represents a Service Destination Point which is a service pipe between routers. Multiple services can utilize an SDP. SDP are unidirectional.

Existence: Will exist in routers supporting a services MIB. The SVC basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
description	text		Description
lspList	CSV		A CSV list of LSP ids
protocol	text		Protocols include: gre and mpls
remoteAddress	IP		The system address of the remote router.
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
uses	child	m-m	Service	A distributed Service uses one or more SDPs



Service

Description: This type represents a Service.

Existence: Will exist in routers supporting a services MIB. The SVC basic extension must be enabled.

Unique Identifier: snmplIndex.

Attributes:

Name	Units	Enum	Description
description	text		Description
isPWSwitching	boolean		Is this Service using pseudowire switching
isRouting	boolean		IS this Service routing
numSAP	int		The number of SAPs associated with this Service
numSDP	int		The number of SDPs this service is using
snmplIndex	OID		See Common Attributes
type	text		Possible values include: epipe, vpls

Relationships

Name	Dir	Card	Participants	Description
access	parent	1-m	SAP	Multiple SAPs can access the same service.
uses	child	m-m	SDP	A distributed Service uses one or more SDPs

ServiceInterface

Description: This type represents an interface used by a SAP. Currently there are no interesting attributes for this object but that should change. Note that if multiple SAP are associated with the same interface, a ServiceInterface object will be created for each SAP. There is no table that represents a ServiceInterface so the snmplIndex is generated by Network Ferret.

Existence: Will exist in routers supporting a services MIB. The SVC basic extension must be enabled.

Unique Identifier: snmplIndex – generated by NF - ifIndex.encapValue (encapValue comes from the SAP).

Attributes:

Name	Units	Enum	Description
ifIndex	int		See Common Attributes
snmplIndex	OID		See Common Attributes

Relationships

Name	Dir	Card	Participants	Description
supports	parent	1-1	SAP	The interface the SAP is using.

8: Advanced Discoveries

This chapter provides common information for the advanced discoveries.

Types of Objects

Connection information will be represented by either Interface objects, Port objects (L2) or some other subclass of `AtiManagedObject` such as `BGPNeighbor` or `OSPFNeighbor`.

There are no relationships in the advanced discovery shipments. Therefore, there will be no diagrams in the following chapters.

Connection IDs

Advanced discoveries do not have relationships in their Shipments. Instead, each side of a connection will be represented in its shipment with enough attribution to indicate the other end. There will also be a **connectionId** attribute to make finding the other end easier.

As of 11.0, all connectionIds are unique. They are generated by the Wire Model. Previously, each advanced discovery would start its connection ID counter from 1.

If no connectionId is present, it means the other end was not found.

For the Tunnel object there will only be a single end. Tunnels are unidirectional. So do not look for a record with a matching connection ID.

Connection Type

All connections, both known and unknown, will have a **connectionType** attribute. The values for this attribute can be found in `AtiFactoryNames.java` near the bottom of the file. Look for:

```
//  
// Advanced discovery connection types  
//
```



Type of Connections

Physical Connections

Physical connections represent a point to point link between two Interfaces/Ports.

Logical Connections

Logical connections represent a logical link between Nodes. Currently, only Router Connectivity (L3) creates logical connections. Logical connections are represented in the shipments by the `AtiManagedObjects` that represent the type of link. Such as `BGPNeighbor` or `OSPFNeighbor`.

Unknown Connections

For a connection where the other end was not found there may be an attribute, **unknownConnection**, that provides information regarding the other end. Unknown connections will **not** have a `connectionId`.

What is Reported

As of 11.0 all connections found via discovery protocols will be reported. Connections found via “guessing” such as bridge forwarding table analysis will NOT be reported if another connection was found.

If a particular shipment contains multiple connections for the same object, that object will be represented once for each connection and each index will be `index`, `index.1`, `index.2`, etc.



9: Switch Connectivity

Remote Information

These attributes can appear in any of the connectivity shipments below. Which attributes appear depends on how much information was gathered about a given remote endpoint.

Name	Units	Enum	Description
remoteAddress	IP		The first pinged IP address associated with the remotePort
remoteAddresses	CSV		The list of IPs associated with the remotePort
remotebplIndex	Text		The bplIndex of the Port that this Port is connected to
z_remoteDiscoveryIP	IP		The discovery IP of the Node of the Port that this Port is connected to. This attribute may or may not have the same value as remoteAddress.
remoteDiscoveryMAC	MAC		The MAC of the discovery IP
remoteIfIndex	int		The ifIndex that the remotebplIndex is related to.
remoteMAC	MAC		The MAC address attached to this port



Switch Shipment

Switches will produce a shipment where the type is SHIPMENT_L2_CONNECTIONS (“**layer2Connections**”) and the node is the discovery IP.

The shipment will consist solely of Port objects.

Description: This represents a connection between a bridge port and another interface/port.

Unique Identifier: bridge port index. Some wireless devices allow multiple connections to the same bridge port. In these cases the unique identifier will be the bridge port index plus a number incrementing from 1.

Attributes

Name	Units	Enum	Description
bplIndex	text		The bridge port index of the port
iflIndex	int		The iflIndex associated with the bridge port index. May not exist.
remoteSysName	text		Only for a link to a DumbRepeater – the name of the DumbRepeater



Dumb Repeater Shipment

Unmanaged hubs will produce a shipment where the type is SHIPMENT_L2_DUMBREPEATER (“**layer2DumbRepeater**”) and the node is the same value as the name of the Node object.

The shipment will contain one Node object which defines the unmanaged hub. Unlike Node objects in Basic Discovery, this Node object will only have a name attribute. The value of the name attribute will be of the form:

```
NonMgdHub.addressOfLayer2Device.1.bridgePortOfLayer2Device
```

where Layer2Device is the managed switch that Network Ferret believes this unmanaged hub is connected to.

The shipment will also contain one or more Port objects. The Port objects which connect back to switches will have the name “uplink#” where # starts at 0 and increments.

An unmanaged hub is created by the MAC matching algorithm, when there is more than one MAC seen on a switch port and those MACs cannot be attributed to another switch.



Port – uplink

Description: This represents a connection from the unmanaged hub to a switch. It is possible that there are links to more than one switch.

Unique Identifier: uplink+x where x starts at 0 and increments

Attributes

Name	Units	Enum	Description
remoteAddress	IP		The IP address of the managed layer-2 device.
remotebplIndex	int		The bplIndex of the Port that this unmanaged hub is connected to
z_remoteDiscoveryIP	IP		The discovery IP of the switch that this unmanaged hub is connected to. This attribute may or may not have the same value as remoteAddress.
remotelfIndex	int		The ifIndex that the remotebplIndex is related to. This may not exist.

Port – host

Description: This represents a MAC address connected to the unmanaged hub.

Unique Identifier: MAC address

Attributes

Name	Units	Enum	Description
remote...			See the table above with all of the remote attributes.

Unconnected IPs Shipment

Not every IP that was discovered will be connected via L2 discovery to switch port. There are many possible reasons for this including having no ARP information, no access to switches, the IP being a loopback IP, etc.

A list of unconnected IPs will be provided via a shipment where the type is SHIPMENT_L2_UNCONNECTEDIPS (“**unconnectedIPs**”) and the node is also SHIPMENT_L2_UNCONNECTEDIPS.

There will be no Node object in the shipment. Neither will there be any relationships. The shipment will consist solely of IP objects.

Any IPs within Layer 2 devices will be ignored as will IPs that were discovered but thrown away due to timeouts or user filters.

IP

Description: This type represents an IP address that was discovered during basic discovery but was not found to be connected to any managed layer-2 device.

Unique Identifier: address

Attributes

Name	Units	Enum	Description
address	IP		The IP address that is not connected.
subnet	IP		The subnet for the address.



Bridge MAC Table Shipment

Production of these shipments must be enabled in the config file.

Each switch will generate a shipment where the type is SHIPMENT_L2_MACS (“**layer2MACs**”).

There will be no Node object in the shipment. Neither will there be any relationships. The shipment will consist solely of Port objects. Each port object will contain an attribute listing the MACs found in the forwarding tables. Note that Network Ferret does a certain amount of cleanup of the MACs it finds in these tables so the MACs reported will not be exactly what can be found in the device.

Port

Description: This type represents a Port that has one or MACs associated with it in the bridge forwarding table.

Unique Identifier: bplIndex

Attributes

Name	Units	Enum	Description
bplIndex	string		The bridge port index of the port
ifIndex	int		The interface index of the port
MAC	list		A comma separated list of MAC addresses that were seen on the port.



VLAN Group Shipments

Switches and routed/bridged VLAN interfaces in routers are grouped by VLAN ID and based on actual connections. So it is possible to have multiple groups with the same VLAN ID when sets of switches are not physically connected.

Each shipment consists of Node objects representing the switches in the group and Interface objects (possibly none) representing the routed/bridged VLAN interfaces.

The shipment type is SHIPMENT_L2_VLANGROUPS ("vlanGroup") and the name of the shipment is VLAN_ID.uniqueID.

Each VLAN ID will report one group with unique ID of 1. This is called the master group and is really for debugging purposes. It contains all of the switches and routed/bridged interfaces with that VLAN ID regardless of connectivity. There will then be one or more shipments with the name VLAN_ID.someNumberOtherThanOne. These are called the sub groups. In the common case where all of the switches and routed interfaces with the same VLAN ID are connected there will be one sub group and it will be identical to the master.

Node

Description: A switch that is a member of the VLAN group. There are no attributes.

Unique Identifier: address

Attributes - none



Interface

Description: A routed/bridged VLAN interface that routes or bridges this VLAN ID and is connected to one of the trunks of the switches in the VLAN group.

Unique Identifier: ifIndex

Attributes

Name	Units	Enum	Description
type	text		The values "route" or "bridge"
z_DiscoveryIP	IP		The discovery address of the router containing the interface



10: Router Connectivity

Router Shipment

Each router produces a shipment whose type is SHIPMENT_L3_CONNECTIONS (“**layer3Connections**”) and whose node value is the discovery IP of the router.



BFDEndpoint

Description: This type represents a BFD relationship between two routers. There is no standard. **These connections are made for Juniper devices only.** RoutingProtocols must be enabled.

Unique Identifier: snmplIndex

Attributes

Name	Units	Enum	Description
localDiscriminator	long		Used to differentiate between multiple links between routers.
remoteAddress	IP		The address of the remote router.
remoteBFDAddress	IP		The remote IP address within the remote reouter
remoteDiscriminator	long		Used to differentiate between multiple links between routers.
snmplIndex	OID		The SNMP index of the local BFDEndpoint.

BGPPeer

Description: This type represents a BGP connection between two routers. Also see the BGP attributes in the basic discovery [Node](#) object. RoutingProtocols must be enabled.

Unique Identifier: snmplIndex (remoteBGPAddress)

Attributes

Name	Units	Enum	Description
localBGPAddress	IP		An address within the local router
localPort	int		IP port number used for communication
remoteAddress	IP		The address Network Ferret used to discover the remote router
remoteBGPAddress	IP		An address within the remote router
remoteBGPAS	int		The remote router's autonomous system number
remoteBGPID	IP		The BGP ID of the remote router.
remotePort	int		IP port number used for communication
snmplIndex	OID		Same as remoteBGPAddress
state	text	yes ->	Translation of the bgpPeerState. See MIB.
version	int		The version of BGP in use



ISISAdjacency

Description: This type represents an ISIS adjacency between two routers.

ISIS interfaces are usually higher level interfaces. The MAC information of the bottom interfaces will be analyzed. If it is determined that these interfaces are directly connected, an Interface connection will also be created between the bottom interfaces. RoutingProtocols must be enabled.

Unique Identifier: snmplIndex

Attributes

Name	Units	Enum	Description
address	IP		An address within the local router
ifIndex	int		The interface to go with the local address
routerId	string		The ISIS ID
remoteAddress	IP		An address within the remote router
remoteRouterIfIndex	int		The router id of the remote router.
remoteRouterId	string		The remote ISIS ID
snmplIndex	OID		



LDPPeer

Description: This type represents an MPLS LDP connection between two routers. It is a combination of the peer and session tables in the LDP MIB. A peering relationship exists when the routers exchange hello packets. A session exists when the routers actually begin communicating. Note that the MPLS MIBs are not settled so support in any given device may exist completely, exist partially, or not exist at all. There are several flavors of MIBs including the standard, RFC-3811, but none will be listed in the table below. MPLS must be enabled.

An LDP session is between Label Spaces so two routers sharing multiple Label Spaces will have multiple connections.

Unique Identifier: snmpIndex

Attributes

Name	Units	Enum	Description
keepAlive	int		The keep alive time on the session. Not always present.
localLDAPAddress	IP		An address within the local router
localLS	OID		The local Label Space
remoteAddress	IP		The address Network Ferret used to discover the remote router
remoteLDAPAddress	IP		An address within the remote router
remoteLS	OID		The remote Label Space
role	text	yes ->	active, passive or unknown. Not always present.
sessionExist	text	yes ->	true, false or unknown. Unknown when MIB not present.
snmpIndex	OID		Same as remoteBGPAddress
state	text	yes ->	Translation of the session state. See MIBs.
targetedPeer	Int	1,2	1=true, 2=false, 0 = not present
version	int		The version of LDP in use



LSP

Description: This type represents an MPLS Label Space Path (LSP). This data is from the experimental MPLS MIB (1.3.6.1.3.95.2.2). MPLS and L3 must be enabled.

The LSP object will be reported in the router shipment for the router that is the head of the LSP. The transit nodes have no interesting information. The tail node may report an ifIndex. If not, the egress ifIndex will be the first loopback interface found on the box or 0 if none.

The values of attributes that exist on all records along the path are taken from the head record.

Unique Identifier: tunnelIndex@ingressIP@egressIP

Attributes

Name	Units	Enum	Description
adminStatus	int		The adminStatus of the LSP
capabilities	Hex string		From the sessionAttributes MIB variable
description	text		
discoveryIP	IP		The discovery IP of the head router
egressIP	IP		The LSR ID of the tail router
ifIndex	int		The ifIndex of the head
ingressIP	IP		The LSR ID of the head router
name	text		
operStatus	int		The operStatus of the LSP
path	list		Comma separated list of IPs along the path. There will always be a head and tail.
remoteDiscoveryIP	IP		The discovery IP of the tail router
remoteIfIndex	int		The ifIndex of the tail. See comment above.



OSPFNeighbor

Description: This type represents an OSPF connection between two routers. Note that if two routers have more than one neighbor relationship, there is no way to tell which one goes with which. They will be connected in the order found. `_RoutingProtocols` must be enabled.

Unique Identifier: `snmpIndex`

Attributes

Name	Units	Enum	Description
<code>localOSPFAAddress</code>	IP		An address within the local router
<code>localOSPFIIndex</code>	int		The interface to go with the local address
<code>remoteAddress</code>	IP		The address Network Ferret used to discover the remote router
<code>remoteOSPFAAddress</code>	IP		An address within the remote router
<code>remoteOSPFIIndex</code>	int		The interface to go with the remote address
<code>remoteRouterId</code>	IP		The router id of the remote router.
<code>snmpIndex</code>	OID		
<code>state</code>	text	yes ->	Translation of the <code>ospfNbrState</code> . See MIB.
<code>version</code>	int		The version of OSPF in use

Interface

Description: This type represents a point to point, physical connection between two router interfaces.

Unique Identifier: `snmpIndex` (`ifIndex`)

Attributes

Name	Units	Enum	Description
<code>connector</code>	boolean		The physical connector value that was reported on the interface in basic discovery.
<code>remoteAddress</code>	IP		The address of the remote router.
<code>remoteSnmpIndex</code>	OID		The SNMP index of the remote Interface
<code>remoteSysname</code>	text		The <code>sysName</code> of the remote router. May not be present.
<code>snmpIndex</code>	OID		The SNMP index of the local Interface



PWVirtualCircuit

Description: This type represents a Pseudo Wire connection between two interfaces. An example of a pseudo wire is a link connecting two switches. As far as the switches are concerned there is a physical wire connecting them. In reality, the “wire” is a PSN (Packet Switched Network) perhaps one supported by MPLS.

There is currently only support for Cisco. Only Ethernet pseudo wires using MPLS and IPv4 are reported

Unique Identifier: snmplIndex (vcIndex)

Attributes

Name	Units	Enum	Description
adminStatus	int		Standard definition of admin status
description	String		The description associated with the local endpoint.
operStatus	int		Standard definition of oper status
PWDERIVEDINDEX	Int		Either a match based on VLAN name, portIfIndex or vcIfIndex
PWPORTIFINDEX	int		Port ifIndex
PWSI	int		See the ServiceInstance type
PWVCID	uns int		A “global” number that MUST be the same for both endpoints.
PWVCIFINDEX	int		VC IF Index
PWVLAN	int		VLAN the vc is running over
remoteAddress	IP		The address of the remote router.
remoteSnmplIndex	int		The internal VC index of the remote endpoint.
snmplIndex	int		The internal VC index of the local endpoint.



Tunnel

Description: This type represents a unidirectional logical tunnel between routers. See the note in the Advanced Discoveries chapter under Connection Ids.

Unique Identifier: snmplIndex

Attributes

Name	Units	Enum	Description
index	Int		The tunnel index (not SNMP index)
metric	Int		The value of the metric variable in the table
nextHopAddress	IP		The NHA may or may not be an address in the remote router.
remoteAddress	IP		The system address of the remote router. This may or may not match NHA.
snmplIndex	OID		The SNMP index of the Tunnel
type	text		Type includes values such as: sdp, ldp, rsvp, bgp

VCEndpoint

Description: This type represents a connection between two Frame Relay DLCIs. These are found via the DLCI polling. Have not seen and in a very long time.

Unique Identifier: snmplIndex

Attributes

Name	Units	Enum	Description
remoteAddress	IP		The address of the remote router.
remoteSnmplIndex	OID		The SNMP index of the remote VCEndpoint
remoteSysname	text		The sysName of the remote router. May not be present.
snmplIndex	OID		The SNMP index of the local VCEndpoint.



11: General Physical Connectivity

This chapter defines the domain model for Connection Discovery. Connection discovery runs before any of the other advanced discoveries since it has no “guessing” algorithms.

Connections Shipment

Each device produces a shipment whose type is SHIPMENT_L3_CONNECTIONS (“**layer3Connections**”) and whose node value is the discovery IP of the device.

If a device (switch) is part of a stack and the connectivity between the switches in the stack is not known, a Node object will be reported in the shipment for each switch in the stack. The shipments for each switch in the stack should have the same set of Node objects. The Node objects will have no attribution.

Interface

Description: This type represents a connection between two device interfaces.

Unique Identifier: snmplIndex (ifIndex)

IMPORTANT NOTE: It is possible that a connection can be found via more than one protocol (i.e. LAG and CDP). Clients wanted all data reported so the first connection will be reported with snmplIndex as snmplIndex. Any subsequent connection will be reported with snmplIndex.# where # starts at 1 and increments. If you do not care how a connection was made, then ignore any snmplIndex containing a dot (.).

Attributes

Name	Units	Enum	Description
remoteAddress	IP		The address of the remote router.
remoteSnmplIndex	OID		The SNMP index of the remote Interface
remoteSysname	text		The sysName of the remote router. May not be present.
snmplIndex	OID		The SNMP index of the local Interface
unknownConnection	text		The CDP name of the other end. The other end device was not discovered.



MaintenanceEndPoint

Description: This type represents Cisco Connectivity Fault Management (CFM) MEPs. These are logical connections but they are special in that they are between interfaces instead of between nodes. We will treat them as a logical connections between two nodes but we will add the interface attribution into the shipment.

Unique Identifier: index (the MEP index which is NOT snmplIndex)

Attributes

Name	Units	Enum	Description
localDirection	string		The direction of the local MEP.
localIfIndex	int		The ifIndex of the local MEP
localMAIndex	int		The index of the local MaintenanceAssociation
localMEPIndex	int		The index of the local MaintenanceEndPoint
MDIndex	int		The index of the MaintenanceDomain
MDLevel	int		The level of the MaintenanceDomain
remoteDiscoveryIP	IP		The address of the remote device.
remoteIfIndex	int		The ifIndex of the remote MEP
remoteMAIndex	int		The index of the remote MaintenanceAssociation
remoteMEPIndex	int		The index of the remote MaintenanceEndPoint
serviceName	string		The service name. The MaintenanceAssociation name.
snmplIndex	OID		The SNMP index of the local Interface



FEX Shipment

One FEX (Fabric Extension) shipment will be created for discovery whose type is SHIPMENT_FEXCONNECTIONS (“**fexConnections**”) and whose node value is also FEXCONNECTIONS.

The shipment will contain all Board objects and REL_MIRRORS relationships.

This is NOT creating connectivity between nodes. This is about FEX boards in servers but another device (Cisco) has necessary information so we must do this analysis after basic discovery.

Board

Description: Either a blade board or an FEX board.

Unique Identifier: Each board’s unique identifier (index).

Attributes

Name	Units	Enum	Description
discoveryIP	IP		The discovery IP of the device the Board is in.
index	text		The unique identifier for the Board.
serial	text		The serial number for the Board.

Relationships

Name	Dir	Card	Participants	Description
mirrors	both	m-1		The FEX board will mirror the blade board. So there can be many relationships to the same blade board.



12: Interface Polling Domain Model

This chapter defines the domain model for Interface Polling which exists as of 8.0.0. See the notes in the Layer 3 chapter.

To make integration of the new discovery easier we use the same shipment name as L3 (Router Connectivity).

Interface Polling is determining connections between aggregated low-level router interfaces.

The Shipment will ONLY contain Interface objects.



Device Shipment

Each device produces a shipment whose type is SHIPMENT_L3_CONNECTIONS (“**layer3Connections**”) and whose node value is the discovery IP of the device.

Interface

Description: This type represents a connection between two device interfaces.

Unique Identifier: snmplIndex (ifIndex)

Attributes

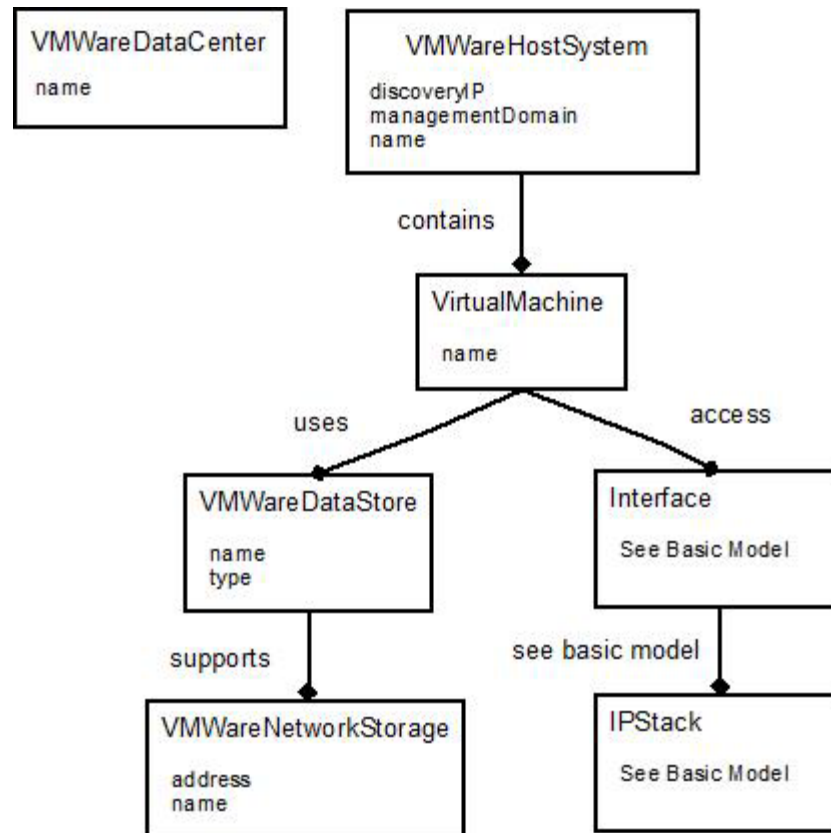
Name	Units	Enum	Description
connector	boolean		Does this interface have a physical connector. Should always be true.
remoteAddress	IP		The address of the remote router.
remoteSnmplIndex	OID		The SNMP index of the remote Interface
snmplIndex	OID		The SNMP index of the local Interface



13: VMWare Domain Model

This chapter defines the domain model for VMWare discovery.

VMWare discovery produces one type of shipment.



DataCenter Shipment

Each VMWare DataCenter produces a shipment whose type is SHIPMENT_VMWARE_DC ("**vmwareDataCenter**") and whose node value is the data center name.

Note that this shipment will also report a standard interface relationship between Interface and IPStack. This represents a network stack used by a Virtual Machine.



VMWareDataCenter

Description: This type represents a VMWare Data Center

Unique Identifier: name

Attributes

Name	Units	Enum	Description
name	Text		The DC name.

VMWareHostSystem

Description: This type represents a VMWare Host System

Unique Identifier: name

Attributes

Name	Units	Enum	Description
cluster	text		The cluster this Host System belongs to, if any
discoveryIP	IP addr		The address of the HostSystem
managementDomain	IP addr		The address of the VCenter managing this host system
name	text		The HS name.

Relationships

Name	Dir	Card	Participants	Description
contains	parent	1-m	VirtualMachine	



VMWareDataStore

Description: This type represents a VMWare Datastore

Unique Identifier: name

Attributes

Name	Units	Enum	Description
name	text		The DS name.
type	text		The DS type. For example, VMFS or NFS.

Relationships

Name	Dir	Card	Participants	Description
supports	Child	1-1	VMWareNetworkStorage	
uses	Child	1-m	VirtualMachine	



VirtualMachine

Description: This type represents a generic Virtual Machine. Some values are only available via the VMWare API.

Unique Identifier: name

Attributes

Name	Units	Enum	Description
capacity	int		The total disk space in bytes that this VM may use. (API only)
description	text		The Annotation object with CR/LF stripped. (API only)
dns	text		The DNS name of the VM. (API only)
dnsIP	IP Addr		The IP address of the DNS name of the VM. (API only)
guestOS	text		Description of the guest OS on this VM.
memory	MB		The memory allocated to this VM
name	text		The VM name.
numCPU	int		Number of CPUs for this VM
snmpIndex	OID		The SNMP Processor only
state	text		The state of the VM as returned by the API (API only)
uuid	text		The VMWare uuid for this VM. Used in VMWare traps

Relationships

Name	Dir	Card	Participants	Description
access	Parent	1-m	Interface	Note that the index of the interface here is the MAC
contains	Child	1-1	VMWareHostSystem	
uses	Parent	1-m	VMWareDataStore	



VMWareNetworkStorage

Description: This type represents Network Storage device with respect to VMWare. Currently the basic Network Ferret model does not contain the concept of Network Storage.

Unique Identifier: name

Attributes

Name	Units	Enum	Description
address	CSV		Comma separated list of IP addresses
name	text		The NS name.

Relationships

Name	Dir	Card	Participants	Description
supports	Parent	1-1	VMWareDataStore	



14: ATM Domain Model

This chapter defines the domain model for ATM discovery. ATM discovery defines a single, simple shipment.

ATM Shipment

ATM connectivity discovery produces a single shipment whose type is SHIPMENT_ATMCONNECTIONS (“**atmConnections**”) and whose node value is also SHIPMENT_ATMCONNECTIONS.

This shipment will contain objects of type Interface and relationships of type atmConnection.

Interface

Description: An ATM interface that is connected to another ATM interface.

Unique Identifier: discoveryIP:ifIndex

Attributes

Name	Units	Enum	Description
node	IP		The IP address of the ATM node this interface is in.
snmplIndex	int		The ifIndex of the interface.
sysName	text		The sysName of the node if it exists.

Relationships

Name	Dir	Card	Participants	Description
atmConnection	both	1-1	Interface, Interface	



15: Handling Non-standard Devices

This chapter discusses standards and how Network Ferret handles devices that do not follow the standards. Since Network Ferret mainly uses SNMP to gather information, this chapter will use SNMP as an example although the discussion is equally valid for other standards.

SNMP Standards

SNMP defines a number of standard MIBs which define the format of the data regarding different objects within the device. Probably the most famous object in networking gear is the Interface and the corresponding interface table in SNMP.

Just about every device in existence supports the interface table. This makes Network Ferret's job very easy because all devices can be handled in the same fashion.

However, SNMP agents inside of devices are written by programmers and sometimes programmers make mistakes and many times programmers must map to the designs of the hardware people. Those designs often do not fit into the current standard because vendors are trying to differentiate their products from the competition. This forces the agent developer to forgo using the standard MIB and implement their own MIB. The general functionality of the feature is the same as the standard but there are many bells and whistles added on that do not fit into the standard MIBs.

There are also times when feature development was faster than the development of the standard MIB for the feature. For example, there are no standard MIBs to define hardware such as shelves, boards and ports. There are no standards for the definition of VLANs. All of this information is stored in private vendor MIBs which are all obviously going to be different from each other.

Common Model

Network Ferret defines a common data model and then maps the vendor-specific information into this model.



The user of Network Ferret receives a consistent data stream and does not need to worry about the vagaries of any particular device.

There are specific points in Network Ferret where code can be inserted to gather vendor-specific information.

The definition of which code will handle which device is codified in files called Vendor Specification files (or VSP files). VSPs are discussed in the next section.

VSP Files

The beginning of this chapter describes the SNMP standard and why many devices do not always follow the standard. Network Ferret uses Vendor Specification files (VSP) to define how Network Ferret should handle a particular device.

Location

VSP files are maintained in the `NFROOT/config/vendor` directory. They are text files that can be edited with any text editor.

Format

VSP files use the same format as the Network Ferret configuration files. Please see the chapter in the User Guide that discusses configuration files.

Inheritance

VSP files inherit attributes from each other. The root VSP file is called root.vsp. It contains attributes that will be common to all VSP files. All other VSP files derive from this file.

The next important VSP file is called SNMPRoot.vsp. Since SNMP is currently the only protocol supported by Network Ferret, SNMPRoot.vsp is the only file that is directly descendent of root.vsp. All other existing VSP files descend from SNMPRoot.vsp.

A VSP defines its parent using the **superclass** attribute. For example, SNMPRoot.vsp has the line:

```
superclass = root
```

The value of **superclass** MUST match the value of **className** in some other VSP file, NOT the name of the file in the file system. To keep things simple, all **className** attributes currently do match the name of the file in the file system. So root.vsp has the line:



`className = root`

Attribute values in a VSP file override any attribute values in an inherited VSP file. So if `root.vsp` contained the line `x=3` and `SNMPRoot.vsp` contained the line `x=4`, `x` would have a value of 4 with respect to `SNMPRoot.vsp` because its values override what is in `root.vsp`.

This is the same concept used in Network Ferret configuration files. The difference is that configuration files only have two levels of inheritance whereas VSP files can have an infinite number of levels of inheritance.

There is a program in the SDK called `AtiVendorSpecificationTester` which will verify that all VSP files are reachable from `root.vsp`.

Customization

If it is required to modify the values in a VSP file that is shipped with Network Ferret, the file should NOT be edited directly. Instead, create a file called `filename.vspx` where `filename` is the same as `filename` of the VSP file to be modified. Put the modifications in this file. Network Ferret always looks for a file called `filename.vspx` after loading a given VSP file.

This makes it easy for a customer to upgrade to a new version of Network Ferret.

If necessary, it is possible to create a `vspx` file that has no corresponding `vsp` file.

VSP File Parameters

All VSP parameters are now documented in `root.vsp` and `SNMPRoot.vsp`.



16: Standard SNMP MIBs and Vendor Support

Below is a listing of SNMP MIBs supported and any vendor-specific variations. In addition to SNMP, Network Ferret also supports the VMWare Web Services API and IP Port scanning.

Many people ask the question, “Is this vendor supported”. The answer is not so simple. If a vendor follows standards and discovers properly, we do not hear about it. We only hear about vendors that do not follow standards, augment standards or have their own special information. So if a vendor is not specifically listed here, it does not mean that a given device does not discover properly.

Basic MIB II

1.3.6.1.2.1

sysDescr	MIB2.1.1.0	Can be used to get OS and version
sysObjectID	MIB2.1.2.0	Used to determine VSP file
sysContact	MIB2.1.4.0	
sysName	MIB2.1.5.0	
sysLocation	MIB2.1.6.0	
sysServices	MIB2.1.7.0	Used to help determine router/bridge
v3EngineId	1.3.6.1.6.3.10.2.1.1.0	

IF-MIB

MIB2.2

ifNumber	IF-MIB.1.0
Interfaces	IF-MIB.2

AT-MIB

MIB2.3

Addr Translation	AT-MIB.1	Pre-ARP
------------------	----------	---------

IP-MIB

MIB2.4

See also:	<u>IPv6-MIB below</u>
ipFwd	IP-MIB.1.0
IPv4 Addresses	IP-MIB.20
NetScaler	Netscaler.4.1.1.26
Timetra	Timetra.3.1.2.3.6.1
Viptela	Viptela.12.2.1
Routing	IP-MIB.21



IPv4 ARP cache	IP-MIB.22
Viptela	Viptela.12.1.1

New Routing	IP-MIB.24.4
ipv6Fwd	IP-MIB.25.0
IPv6 Addresses	IP-MIB.34
IPv6 ARP cache	IP-MIB.35

Transmission-MIB MIB2.10

Frame Relay	Trans-MIB.32.2	For Frame Relay DLCIs
-------------	----------------	-----------------------

Cisco FR	Cisco.9.49
-----------------	------------

MPLS-MIB MIB2.10.166

Label Space	MPLS-MIB.4.1.2.3	
LDP Peers	MPLS-MIB.4.1.3.2	
LSI	MPLS-MIB.2.1.1	
LSIx	1.3.6.1.3.96	Experimental MIB
LSPx	1.3.6.1.3.95	Experimental MIB
VRFI	MPLS-MIB.11.1.2.1	
VRFIx	1.3.6.1.3.118	Experimental MIB
VRF	MPLS-MIB.11.1.2.2	
VRFx	1.3.6.1.3.118	Experimental MIB
VRFRT	MPLS-MIB.11.1.2.3	
VRFRTx	1.3.6.1.3.118	Experimental MIB

Cisco LS	Cisco.10.65.1.2.2	
Cisco LDP Peers	Cisco.10.65.1.3	
Cisco PW	Cisco.10.106.1.2	Pseudo Wire
Cisco EVC	Cisco.9.613.1.4.1	Ethernet Virtual Circuit

Juniper LS	Juniper.3.36.1.2.3	
Jun LDP Peer	Juniper.3.36.1.3.3	
Juniper VRF	Juniper.3.26.1.2	
Juniper VRFI	Juniper.3.26.1.3	VRF Interfaces
Juniper PW	Juniper.3.26.1.4	Pseudo Wire
Juniper RT	Juniper.3.26.1.5	Route Targets

Timetra VRF	Timetra.3.1.2.3.1	
Timetra VRFI	Timetra.3.1.2.3.4	VRF Interfaces

OSPF-MIB MIB2.14

OSPFv3-MIB 1.3.6.1.3.102

Basic Info	OSPF-MIB
------------	----------



Interfaces	OSPF-MIB.7
Neighbors	OSPF-MIB.10

BGP-MIB **MIB2.15**

Basic Info	BGP-MIB
Peers	BGP-MIB.3

Juniper BGP Juniper.5.1.1.2.1.1

Bridge-MIB **MIB2.17**

Ports	Bridge-MIB.1.4	
Spanning Tree	Bridge-MIB.2.15	
Forwarding DB	Bridge-MIB.4.3	
New Fwd DB	Bridge-MIB.7.1.2.2	
VLANs	Bridge-MIB.7.1.4.3	
GVRP	Bridge-MIB.7.1.4.5	GARP VLAN Registration Protocol

Vendor-specific VLAN code

Accelar, Alcatel, Avaya, BayS5000, Centillion, Cisco Kalpana, Cisco, Extreme, HP, Huawei, Juniper, Marconi, Moxa, Netscaler, 3Com

Vendor-specific Spanning Tree and Forwarding DB code

Accelar, Alcatel, BATM (specifically connecting to Cisco), Cabletron 6k, Cabletron MPLUS, Centillion, Cisco (with VLAN indexing), Extreme, HP (via CDP), Marconi, 3Com Lanplex, 3Com CB3500

Other Vendor-specific logic

Extreme Discovery Protocol, Cisco Discovery Protocol

Repeater-MIB **MIB2.22**

Groups	Rptr-MIB.1.2.1	Repeater ports
Addresses	Rptr-MIB.3.3.1	Like the bridge forwarding DB
Mau Status	MIB2.26	

Host Resources-MIB **MIB2.25**

Memory	HR-MIB.2.2
Disk	HR-MIB.3.6
Partition	HR-MIB.3.7
File System	HR-MIB.3.8

See vendor listing under Entity MIB.



EXT-IF-MIB

Interfaces
If Stacking

MIB2.31

EXT-IF-MIB.1.1
EXT-IF-MIB.1.2

ATM-MIB

Neighbors
VPL
VCL
VPx
VCx

MIB2.37

ATM-MIB.1.2
ATM-MIB.6
ATM-MIB.7
ATM-MIB.9
ATM-MIB.11

Marconi

private.353 and private.326

Entity-MIB

Entities
Alias Mapping

MIB2.47

Entity-MIB.1.1.1 Used to determine hardware
Entity-MIB.1.3.2 Used to link to interfaces

Pre Entity-MIB vendor private MIBs

Accelar	2272.1.4	chassis type, card type, ports
Alcatel	800.2	chassis type, card type, ports
APCC	318.1.1.1	UPS objects
Avaya	2167.3.5	chassis type, card type, ports
BATM	738.1.5.5	chassis type
Bay/WellFleet	18.3.4 (Router)	chassis type, card type, ports, OS
Bay/WellFleet	45.1.3.1.7 (S3000)	chassis type, card type
Bay/WellFleet	45.1.6 (S5000)	chassis type, card type, ports
Brocade	1588	OS
Cabletron	52 (Switch)	chassis type, card type
Cabletron	52 (6kSwitch/Card)	Serial
Cabletron	52 (Switch MPLUS)	Multiple switches in same chassis
Centillion	930.2.1.1	card type, ports
Cisco	9 (switches/general +)	chassis type, card type, ports, OS
	1900, 2900, 3k, 5k, 6k, LS1010, Kalpana	
Cisco	9 (routers/rtr cards)	chassis type, card type, ports, OS
Compaq	232	PC adapter, drives, controllers
Compaq	232	PC card, port, OS, CPU, Mem, FS
Dell Pwr Connect	674.10895.3.1	OS name and version
Extreme	1916.1.1	OS version and card types
HP	Switch	OS version out of sysDescr
HP-UX	OS	OS version out of sysDescr
Intel	343.7	chassis type, card type, ports, OS
Juniper	2626.3 (MRouter)	chassis type, card type, ports, OS
Juniper	4874.2.2.2 (UniSphere)	chassis type, card type, ports, OS
Lucent	81 (P333R, P333T)	card type, ports
Lucent	81 (P333R, P333T)	card type, ports



Marconi	326.2.2 (ASX Switch)	chassis type, card type, ports, PS
Marconi	(ES2810)	Uses the Intel MIB
Marconi	326.1.5 (ES3810)	chassis type, card type, ports
MC Data	289.2.1.1.1	chassis type, OS
NetScaler	5951.4.1	OS, CPU, memory, programs, others
Nokia	94	card type, ports
PaloAlto	25461.2.1.2.1	OS, chassis
SilverPeak	none	chassis from ifDescr
SonicWall	none	OS from sysDescr
Sun Mngt Center	42.2.12	disk, file system, CPU, memory
Timetra	6527.3.1.2.2	chassis type, card type, MDAs, OS, PS
3Com	43 (switches)	CB3500, LanPlex, NetBldr, SSI3300
UCD	8072	OS from sysDescr
Unisphere	4874 (Juniper)	OS from sysDescr
Viptela	41916.11.1	OS, chassis, disk, CPU, memory
VMWare	6876	OS type, version, build
Xyplex	33	chassis type

IPv6-MIB

MIB2.55

IPv6 Addresses IPv6-MIB.1.8

VRRP-MIB

MIB2.68

VRRP VRRP-MIB.1.3

Cisco HSRP

Cisco.9.106.1.2.1

IGMP-MIB

MIB2.85

Cache IGMP-MIB.1.2

ISIS-MIB

MIB2.138

Basic Info ISIS-MIB
Interfaces ISIS-MIB.1.3.2
Neighbors ISIS-MIB.1.6.1

Neighbors ISIS-EXP-MIB 1.3.6.1.3.37.1

Huawei

Uses ISIS MIB but slightly different implementation

BFD-MIB

no standard

Juniper

Juniper.5.3.1.1.2.1

CFM-MIB

1.3.111.2.802.1.1.8.1



LLDP-MIB

1.0.8802.1.1.2.1

Remote

LLDP-MIB.4.1

LAG-MIB

1.2.840.10006.300.43

Aggregators

LAG-MIB.1.1.1

Ports

LAG-MIB.1.2.1

ADVA LAG

Different interpretation of the standard MIBs

Cisco LAG

Cisco.9.98.1.1

Foundry LAG

Foundry.1.1.3.33.1.1

Huawei LAG

Huawei.5.25.41.1.3.3

Juniper LAG

Different interpretation of the standard MIBs

Timetra LAG

Timetra.3.1.2.15

Vendor

Not related to any standard

Cisco CDP

Cisco.9.23.1.2.1

Cisco CEF

Cisco.9.492.1.2.2

Foundry FDP

Foundry.1.1.3.20.1.2

Juniper BFD

Juniper.5.3.1.1.2

No BFD standard yet

Timetra SERV

Timetra.3.1.2.4

Services

Viptella SDN

Viptella.4, 5, 6, 12

SD-WAN

