Cisco − OSPF Configuration Management with SNMP

Table of Contents

OSPF Configuration Management with SNMP

Introduction

The Open Shortest Path First (OSPF) routing protocol is defined by RFC 2328 OSPF Version 2 . The goal of this paper is to provide a procedural framework that enables organizations to implement configuration management procedures to verify OSPF deployments against OSPF design plans, and to periodically audit the OSPF deployment to ensure long−term consistency with the intended design.

This paper focuses on the configuration management functions from the ITU−T defined FCAPS (fault, configuration, accounting/inventory, performance, security) model. Configuration management is defined by ITU−T M.3400 as providing functions to exercise control over, identify, collect data from, and provide data to NEs (Network Elements).

The information provided by this paper is presented in several major sections described below.

The OSPF Background section provides a technological overview of OSPF including background information on important aspects of an OSPF deployment.

The Process Definitions section provides an overview of the process definitions used to accomplish OSPF configuration management. The process details are described in terms of goals, performance indicators, inputs, outputs, and individual tasks.

The Task Definitions section provides detailed process task definitions. Each task is described in terms of objectives, task inputs, tasks outputs, resources required to accomplish the task, and job skills needed for a task implementer.

The Data Identification section describes data identification for OSPF. Data identification considers the source of the information or where it is located. For example, information is contained by the system in the Simple Network Management Protocol (SNMP) Management Information Base (MIB), Syslog generated log files, or internal data structures that can only be accessed by the Command Line Interface (CLI).

The Data Collection section of this document describes the collection of the OSPF data. The collection of the data is closely related to the location of the data. For example, SNMP MIB data is collected by several mechanisms such as traps, Remote Monitoring (RMON) alarms and events, or polling. Data maintained by internal data structures is collected by automatic scripts or by a user manually logging into the system to issue the CLI command and then recording the output.

The Data Presentation section provides examples of how the data is presented in report formats. After the data is identified and collected, it is analyzed. This paper provides example reports that may be used to record and compare OSPF configuration data.

The Commercial and Public Internet Monitoring Tools, SNMP Polling Data, and Example Data Collection Algorithms sections provide information on the development of tools to implement the OSPF configuration management procedure.

OSPF Background

OSPF is an internal gateway protocol designed to be used within a single autonomous system. OSPF uses link−state or shortest−path first (SPF)−based technology, as compared to the distance−vector or Bellman−Ford technology found in routing protocols such as Routing Information Protocol (RIP). Individual link−state advertisements (LSAs) describe pieces of the OSPF routing domain, for example, the entire autonomous system. These LSAs are flooded throughout a routing domain, forming the link−state database.

Each router in a domain has an identical link−state database. Synchronization of link−state databases is maintained with a reliable flooding algorithm. From the link−state database, each router builds a routing table by calculating a shortest−path tree, with the root of the tree being the calculating router itself. This calculation is commonly referred to as the Dijkstra algorithm.

LSAs are small and each LSA describes a small piece of the OSPF routing domain, specifically, the neighborhood of a single router, the neighborhood of a single transit network, a single inter−area route, or a single external route.

The following table defines key features of OSPF.

Feature	Description
Adjacency	When pairs of OSPF routers become adjacent, the two routers synchronize their link−state databases by exchanging database summaries in the form of OSPF database exchange packets. Adjacent routers then maintain synchronization of their link−state databases through the reliable flooding algorithm. Routers connected by serial lines always become adjacent. On multi−access networks (Ethernets), all routers attached to the network become adjacent to both the designated router (DR) and the backup designated router (BDR).
Designated router	When a DR is elected on all multi−access networks, it originates the network LSA describing the network's local environment. It also plays a special role in the flooding algorithm, since all routers on the network are synchronizing their link−state databases by sending and receiving LSAs to and from the DR during the flooding process.
Backup designated router	When the current DR disappears, a BDR is elected on multi−access networks to speed the transition of DRs. When the BDR takes over, it does not need to go through the adjacency process on the local−area network (LAN). The BDR also enables the reliable flooding algorithm to proceed in the DR?s absence before the disappearance of the DR is noticed.
Non−broadcast multi−access network support	OSPF treats networks, such as Frame Relay public data networks (PDNs), as if they were LANs. However, additional configuration information is needed for routers attached to these networks to initially find each other.
OSPF configuration management areas	OSPF allows the autonomous systems to be broken up into areas. This provides an extra

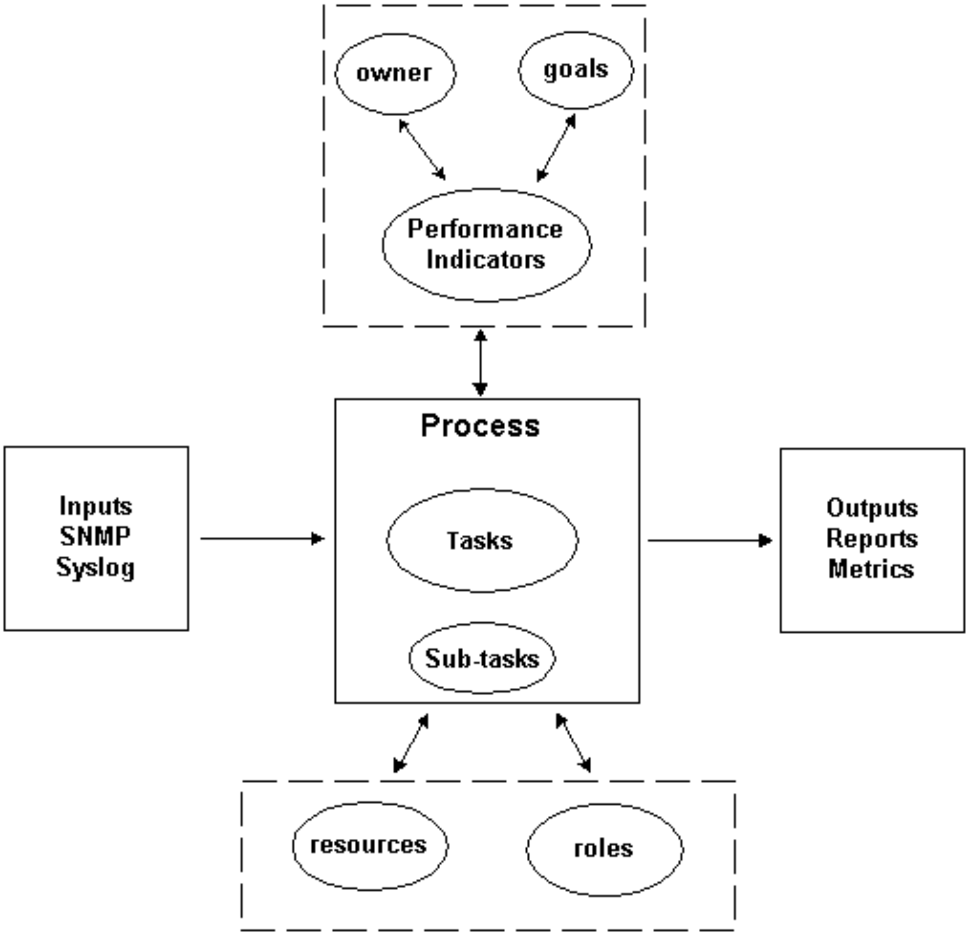
	level of routing protection so that routing within an area is protected from all information external to the area. Also, by splitting an autonomous system into areas, the cost of the Dijkstra procedure, in terms of CPU cycles, is reduced.
Virtual links	By allowing the configuration of virtual links, OSPF removes topological restrictions on area layouts in an autonomous system.
Authentication of routing protocol exchanges	Every time an OSPF router receives a routing protocol packet, it can optionally authenticate the packet before processing it further.
Flexible routing metric	In OSPF, metrics are assigned to outbound router interfaces. The cost of a path is the sum of the path's component interfaces. The routing metric is, by default, derived from the bandwidth of the link. It can be assigned by the system administrator to indicate any combination of network characteristics such as delay, bandwidth, and cost.
Equal−cost multi−path	When multiple best−cost routes to a destination exist, OSPF finds and uses them to load share traffic to the destination.
Variable−length subnet support	Supports variable−length subnet masks by carrying a network mask with each advertised destination.
Stub area support	To support routers having insufficient memory, areas can be configured as stubs. External LSAs are not flooded into and throughout stub areas. Routing to external destinations in stub areas is based solely on default.

Process Definitions

A process definition is a connected series of actions, activities, and changes performed by agents with the intent of satisfying a purpose or achieving a goal.

Process control is the process of planning and regulating, with the objective of performing a process in an effective and efficient way.

Graphically, this is shown in the figure below.



The output of the process has to conform to operational norms that are defined by an organization and are based on business objectives. If the process conforms to the set of norms, the process is considered effective since it can be repeated, measured, managed, and it contributes to the business objectives. If the activities are carried out with a minimum effort, the process is also considered efficient.

Process Owner

Processes span various organizational boundaries. Therefore, it is important to have a single process owner who is responsible for the definition of the process. The owner is the focal point for determining and reporting if the process is effective and efficient. If the process fails to be effective or efficient, the process owner drives the modification of the process. Modification of the process is governed by change control and review processes.

Process Goals

Process goals are established to set the direction and scope for the process definition. Goals are also used to define metrics that are used to measure the effectiveness of a process.

The goal of this process is to provide a framework to verify the deployed configuration of an OSPF implementation against an intended design and to provide a mechanism to periodically audit the OSPF deployment to ensure consistency over time with respect to the intended design.

Process Performance Indicators

Process performance indicators are used to gauge the effectiveness of the process definition. The performance indicators should be measurable and quantifiable. The performance indicators listed below are either numeric or measured by time. Performance indicators for the OSPF configuration management process are defined as follows:

- The length of time required to cycle through the entire process.
- The frequency of execution required in order to proactively detect OSPF issues before they impact users.
- The network load associated with the execution of the process.
- The number of corrective actions recommended by the process.
- The number of corrective actions implemented as a result of the process.
- The length of time required to implement corrective actions.
- The backlog of corrective actions.
- The downtime attributed to OSPF related issues.
- The number of items added, removed, or modified in the seed file. This is an indication of accuracy and stability.

Process Inputs

Process inputs are used to define criteria and prerequisites for a process. Many times, identification of process inputs provide information on external dependencies. A list of inputs related to OSPF configuration management is provided below.

- OSPF design documentation
- OSPF MIB data collected by SNMP polling
- Syslog information

Process Outputs

The process outputs are defined as follows:

- OSPF configuration reports defined in the Data Presentation section of this paper

- OSPF configuration recommendations for corrective actions to be conducted

Task Definitions

The following sections define the initialization and iterative tasks associated with OSPF configuration management.

Initialization Tasks

Initialization tasks are executed once during the implementation of the process and should not be executed with each iteration of the process.

Verify the Prerequisite Tasks

In verifying prerequisite tasks, if it is determined that any one of the tasks is not implemented or does not provide sufficient information to effectively serve the needs of this procedure, this fact should be documented by the process owner and submitted to management. The table below outlines the prerequisite initialization tasks.

Prerequisite Task	Description
Task objectives and inputs	1. Verify that OSPF design documents exist and that the following information is readily available in the network design documentation: 1. Area definitions—Names, address ranges, and area type 2. Area border router/autonomous system border router (ABR/ASBR) identifications 3. DR/BDR identifications 4. Internet Registry (IR) nodes and interfaces assigned to areas 2. Use an SNMP standard configuration template to verify that SNMP is being configured in the network. **Note:** This is used later as input for creating the seed file. 3. Use a Syslog standard configuration template to verify that Syslog is being deployed in the network.
Task output	The task output is a status report on the condition of the prerequisite tasks. If any of the supporting tasks are deemed as ineffective, the process owner should submit a request to have the supporting processes

	updated. If the supporting processes cannot be updated, conduct an assessment on the impact to this process.
Task role	Network engineer skill set.

Create a Seed File

The OSPF configuration management process requires the use of a seed file to remove the need for a network discovery function. The seed file records the set of routers that are governed by the OSPF process and is also used as a focal point to coordinate with the change management processes in an organization. For example, if new nodes are entered into the network, they need to be added to the OSPF seed file. If changes are made to the SNMP community names because of security requirements, those modifications need to be reflected in the seed file. The table below outlines the processes for creating a seed file.

Process	Description
Task objectives	Create a seed file that will be used to initialize OSPF configuration management software. The formatting of the seed file depends on the resources used to implement the OSPF configuration management process. If custom scripts are developed, the format of the seed file is defined by the software design. If a network management system (NMS) is used, the format of the seed file is defined by the NMS documentation.
Task inputs	1. Format the seed file. 2. Use OSPF design documentation to identify the following data: ♦ IP addresses of all nodes ♦ SNMP community strings ♦ Telnet and CLI login accounts and passwords 3. Schedule and/or contact names for the network change management process.
Task outputs	A seed file for the OSPF configuration management process.
Task resources	• Commercial NMS system • Custom developed software system • Manual process—Log into each network element and issue command lines and record the output.
Task role	• NMS—Network engineer, NMS administrator, and NMS script skill sets. • Custom scripts—Network engineer and NMS script skill sets. • Manual processes—Network engineer.

Iterative Tasks

Iterative tasks are executed with each iteration of the process and their frequency is determined and modified in order to improve the performance indicators.

Maintain the Seed File

The seed file is critical for the effective implementation of the OSPF configuration management process. Therefore, the current state of the seed file must be actively managed. Changes to the network that impact the contents of the seed file need to be tracked by the OSPF configuration management process owner.

Process	Description
Task objectives	1. Maintain currency of the seed file through tracking and interactions with organizational functions that control network moves, adds, changes, and/or network configuration modifications. 2. Maintain version control and backup control for the seed file.
Task inputs	1. Information from change management, such as moves, additions, and changes, that impact the content of the seed file. 2. Information from engineering/design that impact the content of the seed file.
Task outputs	1. Weekly report on the status of the seed file currency. 2. Definition and documentation describing the location and restoration procedures for seed file backups.

Execute the OSPF Scan

The two steps used to execute the OSPF scan are:

1. Collecting the data.
2. Analyzing the data.

Depending on how the process is used, the frequency of these two steps will vary. For example, this process can be used to verify installation modifications. In this case, the data collection runs before and after the change, and the data analysis is conducted after the change to determine the success of the change.

If this process is used to verify OSPF configuration management design records, the data collection and analysis frequency is dependent on the rate of change in the network. For example, if there is a significant amount of change in the network, the design verifications are conducted once a week. If there is very little change in the network, the design verifications are conducted no more than once a month.

Review the OSPF Reports

The format of the OSPF configuration management reports is dependent on the resources used to implement the OSPF configuration management process. The following table provides suggested custom developed report formats.

Report	Format
Task inputs	For OSPF configuration management reports, see the Data Presentation section within this document.
Task outputs	If problems are found between the scan reports and the logical design records, a decision must be made as to which item is correct and which is incorrect. The incorrect item should be corrected. This may involve modification of the design records or a network change order.

Data Identification

General Data Characteristics

The following table describes data that can be applied to OSPF configuration management.

Data	Description
OSPF areas	Information that describes the router's attached areas include: • Area ID • Area authentication • SPF runs • Number of ABRs in an area • Number of ASBRs in an area • Area LSA count—Consistency across routers in an area • Area LSA checksum—Consistency across routers in the area • Frequency of packet discards due to addressing errors per area • Frequency of protocol packet discards by the routing process per area • Frequency of routed packet discards due to the *no route found* condition per area
OSPF interfaces	Describes one interface from the viewpoint of OSPF such as: • IP address • Area ID • Administrative status • OSPF metrics assigned to the interface • OSPF timers assigned to the interface • OSPF state

OSPF neighbor state	Describes an OSPF neighbor. • Neighbor router ID • Neighbor state • Neighbor events—The number of times the neighbor relationship has changed state, or an error has occurred. • Neighbor retransmission queue—The current length of the retransmission queue.

SNMP Data Identification

Cisco currently supports the RFC 1253 OSPF Version 2 MIB . RFC 1253 does not contain SNMP trap definitions for OSPF. The latest version of the OSPF MIB is RFC 1850 OSPF Version 2 . SNMP traps are defined for OSPF in RFC 1850. RFC 1850 is not supported on Cisco?s implementation of the OSPF MIB.

Please refer to the SNMP Polling Data section of this document for further details.

Please refer to the Cisco MIB page for a definitive list of which MIBs are supported on which platform and code version.

RMON Data Identification

There is no RMON specific data required for this procedure.

Syslog Data Identification

In general, Syslog generates service−specific messages for different technologies. Although the syslog information is more appropriate for fault and performance management, the information provided here is a reference. For an example of OSPF Syslog information generated by Cisco devices, see OSPF Error Messages.

For a complete list of system messages by facility, please refer to Messages and Recovery Procedures.

Cisco IOS CLI Data Identification

In this version of the OSPF configuration management procedure, there is no CLI data required.

Data Collection

SNMP Data Collection

The table below defines the different components of SNMP data collection.

SNMP Data Component	Definition
General SNMP configuration	For general information on SNMP configuration best practices, see Configuring SNMP.
Service specific SNMP	There are no service−specific SNMP configurations required for this procedure.

configuration	
SNMP MIB requirements	See the SNMP Data Identification section above.
SNMP MIB polling collection	SNMP polled data is collected by a commercial system such as HP OpenView or by custom scripts. For a further discussion of collection algorithms, see the Example Data Collection Algorithms section of this document.
SNMP MIB trap collection	The current version of OSPF MIB supported on Cisco devices does not support SNMP traps. There are no SNMP traps required for this procedure.

RMON Data Collection

There are no RMON configurations and data required in this version of the procedure.

Syslog Data Collection

General syslog configuration guidelines are outside the scope of this document. Please see Configuring and Troubleshooting Syslog for more information.

OSPF specific requirements are addressed by configuring the OSPF router to log neighbor changes with a syslog message using the following command:

```
OSPF_ROUTER(config)# ospf log−adj−changes
```

Cisco IOS CLI Data Collection

In general, the Cisco IOS CLI provides the most direct access to the raw information contained by the NE. However, CLI access is better suited for troubleshooting procedures and change management activities than for global configuration management as defined by this procedure. Access through the CLI will not scale for management of a large network. In these cases, automated information access is required.

In this version of the OSPF configuration management procedure, there are no CLI configurations and data required.

Data Presentation

OSPF Area Report

The following is an example format for the OSPF area report. The format of the report is determined by the capabilities of a commercial NMS, if one is used, or the designed output of the custom scripts.

Area	Data Fields	Last Run	This Run
Area ID #1	Authentication		
	SPF Runs		
	ABR Count		
	ASBR Count		

	LSA Count		
	LSA Checksum		
	Address Errors		
	Routing Discards		
	No Route Found		
Area ID #n	Authentication		
	SPF Runs		
	ABR Count		
	ASBR Count		
	LSA Count		
	LSA Checksum		
	Address Errors		
	Routing Discards		
	No Route Found		

OSPF Interface Report

The following is an example format for the OSPF interface report. In practice, the format of the report is determined by the capabilities of a commercial NMS, if one is used, or the designed output of the custom scripts.

Area	Device	Interface	Data Fields	Last Run	This Run
Area ID #1	Node ID #1	Interface ID #1	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
		Interface ID #n	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
	Node ID #n	Interface ID #1	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
		Interface ID #n	IP Address		
			Area ID		

			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
Area ID #n	Node ID #1	Interface ID #1	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
		Interface ID #n	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
	Node ID #n	Interface ID #1	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		
		Interface ID #n	IP Address		
			Area ID		
			Administration State		
			OSPF State		
			Metrics/Cost/Timers		

OSPF Neighbor Report

The following is an example format for the OSPF neighbor report. In practice, the format of the report is determined by the capabilities of a commercial NMS, if one is used, or the designed output of the custom scripts.

Area	Device	Neighbors	Data Fields	Last Run	This Run
Area ID #1	Node ID #1	Neighbor ID #1	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
		Neighbor ID #n	Router ID		
			Router IP Address		

			State		
			Events		
			Retrans Que		
	Node ID #n	Neighbor ID #1	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
		Neighbor ID #n	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
Area ID #n	Node ID #1	Neighbor ID #1	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
		Neighbor ID #n	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
	Node ID #n	Neighbor ID #1	Router ID		
			Router IP Address		
			State		
			Events		
			Retrans Que		
		Neighbor ID #n	Router ID		
			Router IP Address		
			State		
			Events		

			Retrans Que		

Commercial and Public Internet Monitoring Tools

Commercial tools exist to aid in the collection and processing of syslog information and for the collection polling of general SNMP MIB variables.

No commercial or public Internet monitoring tools are known that support OSPF configuration management as defined by this procedure. Therefore, local custom scripts and procedures are required.

SNMP Polling Data

RFC 1213   Route Table

Object Name	Object Description
ipRouteDest	The destination IP address of the route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table−access mechanisms defined by the network management protocol in use. ::= { ipRouteEntry 1 } object identifier = 1.3.6.1.2.1.4.21.1.1
ipRouteMask	Indicates the mask to be logical with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the corresponding ipRouteDest field belongs to a class A, B, or C network, using one of the following mask networks: • Class A = 255.0.0.0 • Class B = 255.255.0.0 • Class C = 255.255.255.0 If the value of the ipRouteDest is 0.0.0.0, the default route, the mask value is also 0.0.0.0. **Note:** All IP routing subsystems implicitly use this mechanism. ::= { ipRouteEntry 11 } object identifier = 1.3.6.1.2.1.4.21.1.11
ipRouteNextHop	The IP address of the next hop of this route. In the case of a route bound to an interface that is realized with a broadcast media, the value of this field is the agent's IP

	address on the interface. ::= { ipRouteEntry 7 } object identifier = 1.3.6.1.2.1.4.21.1.7
ipRouteIfIndex	The index value that uniquely identifies the local interface through which the next hop of the route is reached. This interface is the same interface identified by the IfIndex value. ::= { ipRouteEntry 2 } object identifier = 1.3.6.1.2.1.4.21.1.2

RFC 1213 Miscellaneous Objects

Object Name	Object Description
ipAdEntIfIndex	The index value that uniquely identifies the interface applicable to the entry. This interface is the same interface identified by the IfIndex value. ::= { ipAddrEntry 2 } object identifier = 1.3.6.1.2.1.4.20.1.2
ipInAddrErrors	The number of input datagrams discarded because the IP address in their IP header was an invalid destination field for the entity. This count includes invalid addresses (0.0.0.0) and unsupported class addresses (class E). For entities that are not IP gateways and do not forward datagrams, the counter includes datagrams discarded because the destination address was not a local address. { ip 5 } object identifier = 1.3.6.1.2.1.4.5
ipRoutingDiscards	The number of valid routing entries discarded. One possible reason for discarding such an entry is to free up buffer space for other routing entries. { ip 23 } object identifier = 1.3.6.1.2.1.4.23
ipOutNoRoutes	The number of IP datagrams discarded because no route could be found to transmit them to their destination. { ip 12 } object identifier = 1.3.6.1.2.1.4.12

RFC 1253 OSPF Area Table

Object Name	Object Description
ospfAreaID	A 32−bit integer uniquely identifying an area. Area ID 0.0.0.0 is used for the OSPF backbone. ::= { ospfAreaEntry 1 } object identifier = 1.3.6.1.2.1.14.2.1.1
ospfAuthType	The authentication type specified for this area. Additional authentication types may be assigned locally on a per−area basis. The default value is 0. ::= { ospfAreaEntry 2 } object identifier = 1.3.6.1.2.1.14.2.1.2
OspfSpfRuns	The number of times the intra−area route table has been calculated using this area's link−state database. object identifier = 1.3.6.1.2.1.14.2.1.4
ospfAreaBdrRtrCount	The total number of ABRs reachable within this area. This is initially 0, the default value, and is calculated in each SPF pass. ::= { ospfAreaEntry 5 } object identifier = 1.3.6.1.2.1.14.2.1.5
ospfASBdrRtrCount	The total number of ABSRs reachable within this area. This is initially 0 (the default value), and is calculated in each SPF pass. ::= { ospfAreaEntry 6 } object identifier = 1.3.6.1.2.1.14.2.1.6
ospfAreaLSACount	The total number of LSAs in an area's link−state database, excluding external LSAs. The default value is 0. ::= { ospfAreaEntry 7 } object identifier = 1.3.6.1.2.1.14.2.1.7
ospfAreaLSACksumSum	The 32−bit unsigned sum of the LSA's LS checksums contained in the area's link−state database. This sum excludes external (LS type 5) LSAs. The sum can be used to determine if there has been a change in a router's link−state database and to compare the link−state database

	of two routers. The default value is 0. ::= { ospfAreaEntry 8 } object identifier = 1.3.6.1.2.1.14.2.1.8

RFC 1253 OSPF Interface Table

Object Name	Object Description
OspfIfIpAddress	The IP address of the OSPF interface. object identifier = 1.3.6.1.2.1.14.7.1.1
OspfIfEvents	The number of times the OSPF interface has changed its state, or an error has occurred. object identifier = 1.3.6.1.2.1.14.7.1.15
OspfIfState	The OSPF interface state. object identifier = 1.3.6.1.2.1.14.7.1.12

RFC 1253 OSPF Neighbor Table

Object Name	Object Description
OspfNbrIpAddr	The IP address of this neighbor. ::= { ospfNbrEntry 1 } object identifier = 1.3.6.1.2.1.14.10.1.1
ospfNbrAddressLessIndex	The corresponding value of IfIndex in the Internet standard MIB on an index that does not have an IP address. On row creation, this can be derived from the instance. ::= { ospfNbrEntry 2 } object identifier = 1.3.6.1.2.1.14.10.1.2
ospfNbrRtrId	A 32−bit integer, represented as an IpAddress, uniquely identifying the neighboring router in the autonomous system. The default value is 0.0.0.0. ::= { ospfNbrEntry 3 } object identifier = 1.3.6.1.2.1.14.10.1.3
ospfNbrState	The state of the relationship with the neighbor. The states are: • down (1) • attempt (2) • init (3)

	• twoWay (4) • exchangeStart (5) • exchange (6) • loading (7) • full (8) ::= { ospfNbrEntry 6 } object identifier = 1.3.6.1.2.1.14.10.1.6
ospfNbrEvents	The number of times the neighbor relationship has changed state, or an error has occurred. The default value is 0. ::= { ospfNbrEntry 7 } object identifier = 1.3.6.1.2.1.14.10.1.7
ospfNbrLSRetransQLen	The current length of the retransmission queue. The default value is 0. ::= { ospfNbrEntry 8 } object identifier = 1.3.6.1.2.1.14.10.1.8

Example Data Collection Algorithms

During the investigation of this paper, a prototype ?C? program was developed. The program, called oscan, was written using Microsoft Developer Studio 97 with Visual C++ version 5.0. There are two specific libraries that provide the SNMP function application programming interface (API). Those libraries are snmpapi.lib and mgmtapi.lib

The functions provided by the Microsoft API are grouped into three major categories and listed in the table below.

Agent Functions	Manager Functions	Utility Functions
SnmpExtensionInit	SnmpMgrClose	SnmpUtilMemAlloc
SnmpExtensionInitEx	SnmpMgrGetTrap	SnmpUtilMemFree
SnmpExtensionQuery	SnmpMgrOidToStr	SnmpUtilMemReAlloc
SnmpExtensionTrap	SnmpMgrOpen	SnmpUtilOidAppend
	SnmpMgrRequest	SnmpUtilOidCmp
	SnmpMgrStrToOid	SnmpUtilOidCpy
	SnmpMgrTrapListen	SnmpUtilOidFree
		SnmpUtilOidNCmp
		SnmpUtilPrintAsnAny
		SnmpUtilVarBindCpy
		SnmpUtilVarBindListCpy
		SnmpUtilVarBindFree
		SnmpUtilVarBindListFree

The oscan prototype code encapsulated the Microsoft API with a set of additional functions listed below.

- snmpWalkStrOid
- snmpWalkAsnOid

- snmpWalkVarBind
- snmpWalkVarBindList

These functions provide a generic API that allow access to the various SNMP MIB tables used to maintain the OSPF configuration data. The object identifier (OID) for the table to be accessed is passed to the oscan API along with a table specific call back function. The call back function has the intelligence to act on the data returned from the tables.

Main Routine

The first task is building a list of nodes that will be the target of the oscan program. In order to avoid the ?device discovery? problem, a seed file is required to identify the nodes to be scanned. The seed file provides information such as the IP address and the SNMP read−only community strings.

The oscan program needs to maintain several internal data structures to store the SNMP information that is collected from the routers. In general, there is an internal data structure for each SNMP MIB table that is collected.

```
Main
        load node array based on information in the seed file.
        while more entries in the node array
                start SNMP session for this node
                collect IP route table for this node
                collect OSPF area table for this node
                collect OSPF Neighbor table for this node
                collect  sysName for this node
                collect OSPF Interface table for this node
                end SNMP session for this node
        end while
```

IP Route Table

Care must be taken while accessing the IP route table with SNMP since it is simple to overload a router?s CPU during this operation. Therefore, the oscan program utilizes a user configurable delay parameter. The parameter provides a delay between each SNMP request. For large environments, this means that the total time to collect the information can be very significant.

The route table contains four pieces of information that oscan is interested in:

- ipRouteDest
- ipRouteMask
- ipRouteNextHop
- ipRouteIfIndex

The route table is indexed by ipRouteDest. Therefore, each object that is returned from the SNMP **get−request** has the ipRouteDest appended to the OID.

The object ipRouteIfIndex is an integer that indexes into the IP address table (ipAddrTable). The ipAddrTable is indexed using the ipAdEntAddr object (the IP address of the interface). In order to get the IP address of the interface, a four−step process is required:

1. Collect the ipRouteIfIndex from the routing table.
2. Access the ipAddrTable using the ipRouteIfIndex for pattern matching.

3. When a pattern is found, convert the OID to a string and collect the last four dotted decimal fields that will be the IP address of the interface.
4. Store the IP address of the interface back into the IP route table.

The general algorithm for accessing the IP route table is shown below. At this point, only the integer value of the ipRouteIfIndex is stored. Later in the process, when collecting the interface information, the ipAddrTable is accessed and the remaining information is collected and placed into the internal IP route table.

```
OID List =
        ipRouteDestOID,
        ipRouteMaskOID,
        ipRouteNextHopOID,
        ipRouteIfIndexOID;

For each object returned by  SNMP route table walk
        Sleep // user configurable polling delay.
        check varbind oid against OID list
        if OID is ipRouteDestOID
                add new entry in the internal route table array
        if OID is one of the others
                search internal route array for matching index value
                store information in array
```

The information collected is represented in a table that resembles the familiar output from the router CLI below.

```
ROUTE TABLE
**********************************************************
Destination     Mask                GW              Interface
10.10.10.4      255.255.255.252     10.10.10.5      10.10.10.5
10.10.10.16     255.255.255.252     10.10.10.6      10.10.10.5
10.10.10.24     255.255.255.252     10.10.10.25     10.10.10.25
10.10.10.28     255.255.255.252     10.10.11.2      10.10.11.1
10.10.10.36     255.255.255.252     10.10.10.6      10.10.10.5
10.10.11.0      255.255.255.0       10.10.11.1      10.10.11.1
10.10.13.0      255.255.255.0       10.10.11.2      10.10.11.1
```

OSPF Area Table

Collection of information from the OSPF area table is done by scanning the OSPF area table (ospfAreaTable) and processing the data as it is returned. The index to the ospfAreaTable is the osfpAreaId. The ospfAreaId is stored in dotted decimal format which is identical to an IP address. Therefore, the same subroutines that were used to process and search for the ipRouteTable and ipRouteIfIndex can be re−used here.

There are several data items that are not actually in the OSPF area table that are included in this section. For example, the ipInAddrErrors, IpRoutingDiscards, and ipOutNoRoute objects are in the MIB−2 definition, but are not associated with an OSPF area. These objects are associated with a router. Therefore, these counters are used as an area metric by adding the values for each node in an area to an area counter. For example, in the OSPF area report, the number of packets discarded due to *no route found* is actually the sum of the packets discarded by all of the routers in that area. This is a high level metric that provides a general view of the routing health of the area.

```
        OID List =
        ipInAddrErrorsOID,
        ipRoutingDiscardsOID,
        ipOutNoRouteOID,
        areaIdOID,
        authTypeOID,
        spfRunsOID,
        abrCountOID,
        asbrCountOID,
        lsaCountOID,
        lsaCksumSumOID;

        For object returned from the SNMP walk of  the Area Table
                Sleep // user configurable polling delay.
                check varbind oid against OID list.
                if OID is ospfAreaId
                        add new entry in the internal route table array
                if OID one of the others
                        search internal array for matching index value
                        store information in array
        end of for loop
        get ipInAddrErrors, ipRoutingDiscards, ipOutNoRoute
        add values to overall Area counters
```

The information collected is represented in the ASCII table below.

```
AREAS
**********************************************************
AREA = 0.0.0.0                  AREA = 0.0.0.2
authType = 0                    authType = 0
spfRuns = 38                    spfRuns = 18
abrCount = 2                    abrCount = 1
asbrCount = 0                   asbrCount = 0
lsaCount = 11                   lsaCount = 7
lsaCksumSum = 340985            lsaCksumSum = 319204
ipInAddrErrors = 0                 ipInAddrErrors = 0
ipRoutingDiscards = 0           ipRoutingDiscards = 0
ipOutNoRoutes = 0               ipOutNoRoutes = 0
```

OSPF Neighbor Table

The index for the neighbor table is two values:

- ospfNbrIpAddr—The ospfNbrIpAddr is the IP address of the neighbor.
- ospfNbrAddressLessIndex—The ospfNbrAddressLessIndex can be one of two values:
    - For an interface that has an IP address assigned, it is zero.
    - For an interface that does not have an IP address assigned, it is interpreted as the IfIndex from the Internet standard MIB.

Because there are two values for the index, you need to adjust the algorithms used earlier for the extra information appended to the returned OIDs. After making this adjustment, the same subroutines that were used to process and search for the ipRouteTable and ipRouteIfIndex can be re−used here.

```
OID List =
ospfNbrIpAddrOID,
ospfNbrAddressLessIndexOID,
ospfNbrRtrIdOID,
ospfNbrStateOID,
ospfNbrEventsOID,
ospfNbrLSRetransQLenOID,

For object returned from the SNMP walk of the Neighbor Table
```

```
                Sleep // user configurable polling delay.
                check varbind OID against OID list.
                if OID matches ospfNbrIpAddr
                        add new entry in the internal neighbor table array
                if OID matches one of the others
                        search array for matching index value
                        store information in array
```

The information collected is represented in the ASCII table below.

```
NEIGHBORS
************************************************************
NEIGHBOR #0                             NEIGHBOR #1
Nbr Ip Addr = 10.10.10.6                Nbr Ip Addr = 10.10.11.2
Nbr Rtr Id = 10.10.10.17                Nbr Rtr Id = 10.10.10.29
Nbr State = 8                           Nbr State = 8
Nbr Events = 6                          Nbr Events = 30
Nbr Retrans = 0                         Nbr Retrans = 0
```

Related Information

Related Topics

- **OSPF Design Guide Section 1**
- **OSPF Design Guide Section 2**
- **OSPF Technical Tips and IP Routing Top Issues**
- **OSPF Database Explanation Guide**
- **OSPF Configuration Guide**
- **Best Practices White Papers**

Additional Topics

- **Performance Management: Cisco NSA HAS Best Practice White Paper**
- **Capacity and Performance Management: Cisco NSA HAS Best Practice White Paper**
- **Configuration Management: Cisco NSA HAS Best Practice White Paper**
- **Change Management: Cisco NSA HAS Best Practice White Paper**
- **RFC 1246 Experience with the OSPF Protocol**
- **RFC 1245 OSPF Protocol Analysis**

- **RFC 1224 Techniques for Managing Asynchronously Generated Alerts**

Additional Reading

- Thomas II, Thomas. *OSPF Network Design Solutions*. Cisco Press; 1998; ISBN 1578700469
- Maggiora, Paul Della. *Performance and Fault Management*. Cisco Press; 2000; ISBN 1578701805

Updated: May 02, 2002 Document ID: 15408