EMC® Smarts® IP Management Suite

7.0

Discovery Guide

P/N 300-004-087
REV A02

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Copyright © 1996-2007 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com

All other trademarks used herein are the property of their respective owners.

Contents

Preface ix

Chapter 1 Introduction

Important terminology 1-2
What do system and device mean? 1-2
What is a Domain Manager? 1-2
What is discovery? 1-2
Supported network systems and discovery protocols 1-3
Discovery overview 1-4
IP Availability Manager Light Discovery 1-5
IP Availability Manager Extension for IPv6 Discovery 1-7
IP Availability Manager Extension for NAS Discovery 1-7
Discovery and IP tagging 1-8
Turning on VRF IP tagging 1-9
Using IP tagging 1-10
IPSec discovery 1-12
Load balancer discovery 1-13

Chapter 2 Discovery Process

Discovery methods 2-2
Manual discovery 2-2
Autodiscovery 2-2
Topology import 2-2
Discovery phases 2-5
Phase 1: Perform initial polling of the candidate system 2-5
Phase 2: Determine the destination of the candidate system 2-6
Phase 3: Probe the system added to the topology 2-8
Phase 4: Post-process the discovery information 2-18
Autodiscovery data sources 2-19
Topology MIBs data source 2-19
IP address table MIB data source 2-19
ARP table MIB data source 2-19
Short discovery for IPSec tunnels 2-20
Monitoring and analysis 2-20
Short discovery characteristics 2-20

Chapter 3 Name Resolution

Discovery and the local name resolution service 3-2
Hostname resolution and IP address checking..... 3-3
How discovery names a system..... 3-4
TM_USEAUTONAME option for automatic name resolution 3-4
TM_USESEEDME option for seed-file name resolution 3-5
Local exceptions to automatic and seed-file name resolution..... 3-5
Configuring automatic name resolution 3-6
Configuring exceptions to automatic name resolution 3-8
Configuring seed-name name resolution..... 3-9
Configuring exceptions to seed-name name resolution..... 3-10

Chapter 4 Preparing and Initiating Discovery

Preparing for manual discovery 4-2
Preparing for autodiscovery..... 4-3
Create a seed file to seed the topology..... 4-3
Create one or more discovery filters 4-3
Specify autodiscovery safeguards 4-3
Select autodiscovery data sources 4-5
Enable autodiscovery 4-6
Preparing for topology import..... 4-7
Initiating discovery 4-8
Using the import from seed file command to initiate discovery 4-8
Using the Add Agent command to initiate discovery..... 4-8
Initiating discovery for topology import..... 4-11
Stopping autodiscovery 4-12

Chapter 5 Opening the Global Console

Opening the Global Console..... 5-2
User accounts and passwords..... 5-2
Procedure for opening the Domain Manager Administration Console . 5-2
Layout of the Domain Manager Administration Console 5-4
Domain Manager Administration Console toolbar buttons..... 5-5

Chapter 6 Preparing a Seed File

About seed files 6-2
Seed file template 6-2
Seed file dual use..... 6-2
Seed file format..... 6-3
Notation and syntax 6-5
Specifying a port 6-5
Specifying a community string 6-6
Mandatory keyword options for SNMPv1 or v2c..... 6-6
Mandatory keyword options for SNMPv3..... 6-6
Seed file examples for v1/v2c candidate systems..... 6-8
Seed file examples for v3 candidate systems 6-9
What systems to put in a seed file 6-10
Seed file sources 6-10
Autodiscovery seed systems 6-10
SNMPv3 systems..... 6-11
Multiple seed entries for multiple IP-address systems 6-11

How to create a seed file 6-12
How to encrypt passwords in the seed file 6-13

Chapter 7 Creating Discovery Filters

About discovery filters 7-2
Discovery filter fields 7-3
Extended filter information 7-4
Examples of discovery filters 7-7
Filtering a range of IP addresses 7-7
Filtering by system type, name, and description 7-7
Filtering with manual accept mode 7-8
Filter for EMC Smarts IP adapter 7-8
Filtering by system OID 7-9
How to create a discovery filter 7-10

Chapter 8 Understanding Discovery Results

Discovery results 8-2
Successful discovery 8-2
Unsuccessful discovery 8-2
Unsuccessful rediscovery 8-2
Discovery error scenarios and results 8-3
Discovery errors 8-5
Discovering a system hostname that does not resolve to an IP address 8-5
SNMP agent violates SNMP protocol 8-6
System does not support SNMP 8-7
Incorrect read community string 8-7
Insufficient number of volume licenses 8-7
DiscoveryError notifications 8-9
SNMP request times out 8-9
SNMP agent loops 8-9
System Down 8-10
Qualified access address not found 8-10
System previously discovered fails authentication 8-10
Explanation modes differ 8-11
Pending Devices list 8-12
Information provided by Pending Devices list 8-13
Managing systems on the Pending Devices list 8-14
Fine tuning parameter settings in discovery.conf 8-15

Chapter 9 Customizing Discovery

Certifying non-network systems 9-2
Using discovery hook scripts 9-4
When the discovery hook scripts run 9-4
Discovery hook script files 9-4
Modifying a discovery hook script file 9-5
Example discovery hook script file 9-5

Chapter 10 Scheduling and Manually Invoking Discovery

Automatic and manual discovery 10-2
Full discovery 10-2
Pending discovery 10-2
Discovery considerations 10-3

Scheduling automatic discovery 10-4
Configure the full discovery interval 10-4
Configure the pending discovery interval 10-5
Invoking discovery from the console 10-6
Invoking discovery from the command line 10-7

Chapter 11 Working with the Topology

Adding and removing read community strings 11-2
How to add read community strings 11-2
How to delete read community strings 11-3
How to specify additional read community strings 11-3
Managing and unmanaging topology objects 11-4
Determining the managed status of an object 11-5
Types of objects that can be managed or unmanaged 11-5
Rules governing manage and unmanage 11-5
Managing and unmanaging individual objects 11-6
Persistent management state 11-6
Duplicate IP address errors 11-7
Extracting a seed file from the IP Manager 11-8
Removing systems from the topology 11-9
Saving the topology 11-10

Appendix A ICIM Network Concepts

ICIM overview A-2
ICIM models chart A-3
Two ICIM hierarchies A-4
The physical hierarchy A-4
The logical hierarchy A-4
Joining the two hierarchies A-9
More information about object types A-10
Unitary computer systems and management agents A-10
Chassis and cards A-11
Network adapters and device connections A-12
Protocol endpoints and logical links A-14
Redundancy groups A-14
HSRP groups A-15
Topology examples A-16
Host connected to a switch A-16
Wide area network A-16
Switch with routing module A-17
ICIM instrumentation classes A-19

Appendix B Polling Used During Discovery

What is ICMP? B-2
IPv4 ICMP poller B-2
IPv6 ICMP poller B-2
What is SNMP? B-3
SNMP key components B-3
SNMP basic operations B-4
SNMP versions B-4
EMC Smarts SNMP version support B-5
Controlling EMC Smarts ICMP and SNMP polling B-6

Appendix C Wildcard Patterns

Types of wildcard patterns C-2

Index i-1

Preface

The EMC Smarts IP Management Suite is a family of products intended to diagnose availability and performance problems for IP networks.

This document provides an overview of the discovery process for the EMC Smarts IP discovery products and presents procedures for initiating discovery. The IP products that provide discovery are:

- EMC Smarts IP Availability Manager (IP Availability Manager)
- EMC Smarts IP Performance Manager (IP Performance Manager)
- EMC Smarts IP Server Performance Manager (IP Server Performance Manager)
- EMC Smarts IP Discovery Manager (IP Discovery Manager)

IP Availability Manager discovers and monitors network data for the purpose of performing availability analysis, and IP Performance Manager and IP Server Performance Manager discover and monitor network data for the purpose of performing performance analysis. IP Discovery Manager has the same discovery capabilities as IP Performance Manager except that it does not perform any availability or performance analysis.

Note: In this document, "IP Manager" is used to collectively represent IP Availability Manager, IP Performance Manager, IP Server Performance Manager, and IP Discovery Manager.

Audience for this guide

This document is intended for IT managers seeking to understand how the discovery process works, and for system administrators responsible for the administration, configuration, or use of the IP Manager to discover IP network systems.

Prerequisites

Before you perform the procedures in this document, the following software must be installed:

- EMC Smarts IP Availability Manager, EMC Smarts IP Performance Manager, EMC Smarts IP Server Performance Manager, or EMC Smarts IP Discovery Manager
- EMC Smarts Service Assurance Manager (Global Manager)
- EMC Smarts Global Console

The *EMC Smarts IP Management Suite Installation Guide* and the *EMC Smarts Service Assurance Management Suite Installation Guide* provide information about installing these products.

It is also assumed that the software has been configured in accordance to the *EMC Smarts IP Management Suite Configuration Guide*.

EMC Smarts IP Management Suite installation directory

In this document, the term BASEDIR represents the location where EMC Smarts software is installed.

- For UNIX, this location is: /opt/InCharge*<n>*/*<productsuite>*.
- For Windows, this location is: C:\InCharge*<n>**<productsuite>*.

The <n> represents the EMC Smarts software platform version number. The <productsuite> represents the EMC Smarts product suite to which the product belongs. For example, on UNIX operating systems, EMC Smarts IP Availability Manager is, by default, installed to: /opt/InCharge7/IP/smarts. On Windows operating systems, this product is, by default, installed to: C:\InCharge7\IP\smarts. This location is referred to as BASEDIR/smarts.

Optionally, you can specify the root of BASEDIR to be something different, but you cannot change the <productsuite> location under the root directory.

For more information about the directory structure of EMC Smarts software, refer to the *EMC Smarts System Administration Guide*.

EMC Smarts IP Management Suite products

The EMC Smarts IP Management Suite includes the following products:

- EMC Smarts IP Availability Manager
- EMC Smarts IP Discovery Manager
- EMC Smarts IP Performance Manager
- EMC Smarts IP Server Performance Manager
- EMC Smarts Adapter for HP OpenView NNM
- EMC Smarts Adapter for IBM/Tivoli NetView

Related documentation

In addition to this document, EMC Corporation provides a Help system for command line programs as well as product documentation.

Help for command line programs

Descriptions of command line programs are available as HTML pages. The index.html file, which provides an index to the various commands, is located in the BASEDIR/smarts/doc/html/usage directory.

EMC Smarts documentation

Readers of this guide may find the following related documentation helpful. It can be found in the BASEDIR/smarts/doc/pdf directory.

Note: These documents are updated periodically. Electronic versions of the updated manuals are available on the Powerlink website:
http://Powerlink.EMC.com.

- *EMC Smarts Documentation Catalog*
- *EMC Smarts System Administration Guide*
- *EMC Smarts ICIM Reference*
- *EMC Smarts Common Information Model Infrastructure Models Chart*
- *EMC Smarts Common Information Model Application/Business Models Chart*
- *EMC Smarts ASL Reference Guide*
- *EMC Smarts Perl Reference Guide*
- *EMC Smarts Dynamic Modeling Tutorial*
- *EMC Smarts MODEL Reference Guide*

EMC Smarts IP Management Suite documentation

The following EMC Smarts documents are relevant to users of the IP Management product suite:

- *EMC Smarts IP Management Suite Installation Guide*
- *EMC Smarts IP Management Suite Deployment Guide*
- *EMC Smarts IP Management Suite Configuration Guide*
- *EMC Smarts IP Management Suite Discovery Guide*
- *EMC Smarts IP Availability Manager User's Guide*
- *EMC Smarts IP Performance Manager and Server Performance Manager User's Guide*
- *EMC Smarts IP Adapters User's Guide*
- *EMC Smarts IP Availability Manager Extension for IPv6 User's Guide*
- *EMC Smarts IP Availability Manager Extension for NAS User's Guide*
- *EMC Smarts Quick Start Guide*

Refer to the *EMC Smarts Documentation Catalog* for documentation resources provided with other EMC Smarts product suites.

Suggestions for searching PDF files

You may search across multiple PDF files using the Adobe Acrobat Reader software.

1. If the documentation is not accessible to all users of the EMC Smarts product suite, copy the contents of the *BASEDIR/smarts/doc/pdf d*irectory to a central location, such as a shared drive on your LAN, that operators and others may view.
2. To search throughout the documentation library, open the Acrobat Reader software.
   1. Choose **Edit** > **Search**, and enter a word or phrase.
   2. In the **Where would you like to search** option, select **All PDF Documents in** and type the pathname of the location where the PDF documents reside.

If you have more than one EMC Smarts product suite installed, you can set up cross-product document searches by copying files from the BASEDIR/smarts/doc/pdf directory for each product suite into this common documentation directory path.

Conventions used in this guide

EMC uses the following conventions for notes and cautions.

Note: A note presents information that is important, but not hazard-related.

IMPORTANT

An important notice contains information essential to the operation of the software.

CAUTION

A caution contains information essential to avoid data loss or damage to the system or equipment. The caution may apply to hardware or software.

Typographical conventions

EMC uses the following conventions in this guide:

Normal font	In running text: • Interface elements (for example, button names, dialog box names) outside of procedures • Items that the user selects outside of procedures • Java classes and interface names • Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, filenames, functions, menu names, utilities • Pathnames, URLs, filenames, directory names, computer names, links, groups, service keys, file systems, environment variables (for example, command line and text), notifications
Bold	In procedures: • Names of dialog boxes, buttons, icons, menus, fields, and maps • Selections from the user interface, including menu items, and field entries • Key names • Window names • Code examples • Commands • User interface tabs
Italic	Used for: • Full publications titles referenced in text In procedures: • Directory paths • Filenames • Scripts • Important/unique words • URLs • Utilities • Variables
`Courier`	Used for: • System output • User entry
< >	Angle brackets used to enclose parameter or variable values supplied by the user.
[ ]	Square brackets used to indicate optional values.
|	Vertical bar used to indicate alternate selections. The bar means “or”.

{ }	Braces used to indicate content that you must specify (that is, x or y or z).
...	Ellipses used to indicate nonessential information omitted from the example.
%	Used to indicate a C shell prompt.
#	Used to indicate a C shell superuser prompt.
▼▲	Used to indicate a command is wrapped over one or more lines. The command must be typed as one line.

Pathname conventions

Directory pathnames are shown with forward slashes (/). Users of the Windows operating systems should substitute back slashes (\) for forward slashes.

Graphical conventions

If there are figures illustrating consoles in this document, they represent the consoles as they appear in Windows. Under UNIX, the consoles appear with slight differences. For example, in views that display items in a tree hierarchy such as the Topology Browser, a plus sign displays for Windows and an open circle displays for UNIX.

Smarts Manager

Unless otherwise specified, the term Smarts Manager is used to refer to EMC Smarts programs such as Domain Managers, Global Managers, and adapters.

Where to get help

EMC support, product, and licensing information can be obtained as follows.

Product information — For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Powerlink™ website (registration required) at:

http://Powerlink.EMC.com

Technical support — For technical support, go to EMC Customer Service on Powerlink. To open a service request through Powerlink, you must have a valid support agreement. Please contact your EMC sales representative for details about obtaining a valid support agreement or to answer any questions about your account.

Sales and customer service contacts — For the list of EMC sales locations, please access the EMC home page at:

http://EMC.com/contact

1

Introduction

This chapter introduces the IP Manager and describes the concepts of discovering network topology using the IP Manager. It consists of the following sections:

- Important terminology 1-2
- Supported network systems and discovery protocols 1-3
- Discovery overview 1-4
- Discovery and IP tagging 1-8
- IPSec discovery 1-12
- Load balancer discovery 1-13

Important terminology

In this document, “IP Manager” collectively represents EMC® Smarts® IP Availability Manager (IP Availability Manager), EMC Smarts IP Performance Manager (IP Performance Manager), EMC Smarts IP Server Performance Manager (IP Server Performance Manager), and EMC Smarts IP Discovery Manager (IP Discovery Manager).

Note: IP Server Performance Manager is an enhanced version of IP Performance Manager and is available through a feature license.

What do system and device mean?

The term "system," as used in this document, is a generic term representing a computer-based network entity, such as a switch, router, hub, bridge, or host. The term "device," as used in this document, has essentially the same meaning as system except that "device" also conveys a sense of specific model, such as a specific model of switch, router, hub, bridge, or host.

What is a Domain Manager?

IP Availability Manager, IP Performance Manager, and IP Server Performance Manager are examples of EMC Smarts Domain Managers, where a Domain Manager is a service-assurance application associated with a particular type of information-technology domain (networks, systems, applications, application services). For IP Availability Manager, IP Performance Manager, and IP Server Performance Manager, the domain is IP network transport. Each Domain Manager is autonomous in the sense that it maintains its own data models, repository, and problem signatures, and monitors and analyzes the discovered objects in its own domain.

IP Discovery Manager performs the same discovery as IP Performance Manager, but it does not perform any analysis on the discovered network objects. Its purpose is to provide a database of discovered network objects and their connectivity to any application requiring such information.

What is discovery?

Stated simply, EMC Smarts discovery is the process of creating a representation of the managed topology within the repository, or database, of a Domain Manager. For the IP Manager, data is collected to create instances of the managed network systems (switches, routers, hubs, bridges, hosts), their dependencies, their internal components, and their connections.

When a system (device) is added to the managed network environment, the IP Manager performs discovery to determine the system’s configuration and its relationships to other managed systems and components. When a system is removed from the managed network environment and deleted from the topology, the IP Manager removes the system and all of its components from the modeled topology.

Systems can be added to or deleted from the modeled topology at any time. Adding new systems does not impede the ability of the IP Manager to monitor systems that are already managed.

Supported network systems and discovery protocols

The IP Manager discovers and models the following types of network systems:

- Bridges
- Firewalls
- Hosts
- Hubs
- Load balancers
- Probes
- Routers
- Router cards
- Switches
- Terminal servers

Appendix A, "ICIM Network Concepts," provides information about the classes used by the IP Manager to model these systems, and the *EMC Smarts 7.0 Certified Device List* provides information about the vendor-specific models supported by the IP Manager for these systems.

The IP Manager uses the ICMP and SNMP protocols to discover the systems in the managed network environment. Appendix B, "Polling Used During Discovery," provides information about ICMP and SNMP.

Discovery overview

IP Availability Manager, IP Performance Manager, IP Server Performance Manager, and IP Discovery Manager run variations of the same discovery manager program. The configurations and licenses applied to an IP Manager at startup determine which discovery the product will run.

The IP Manager discovers Layer 2 (data-link) and Layer 3 (network) connectivity in multi-vendor, switched, and routed networks. (Layer 2 network systems transmit packets from network node to network node based on machine address, and layer 3 network systems route data to different LANs and WANs based on network address.) The IP Manager discovers the network systems by sending them ICMP and SNMP polls. (See Figure 1-1.)

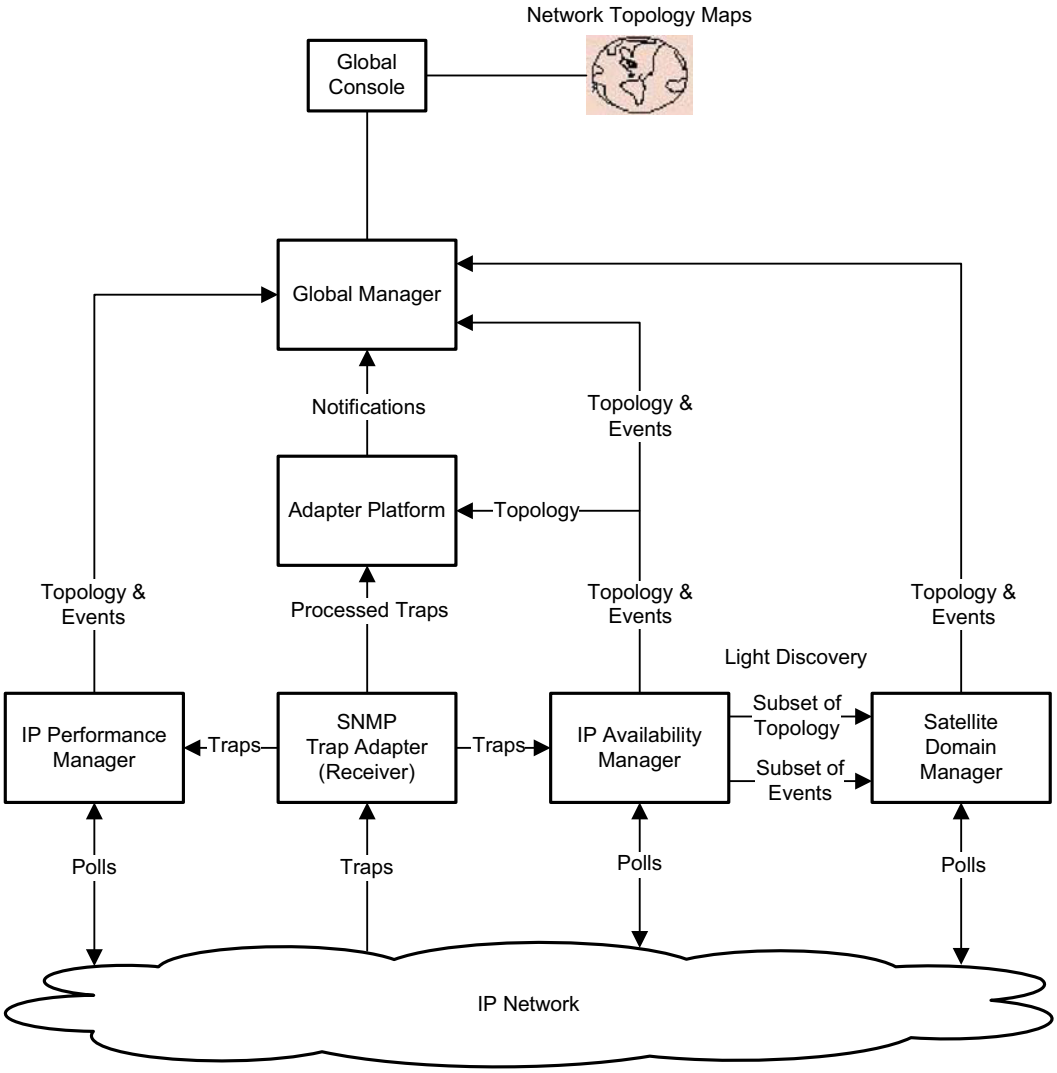


Figure 1-1 **IP Manager discovery**

Note: In Figure 1-1, running separate instances of IP Availability Manager and IP Performance Manager is necessary if the two applications are running on different hosts. When running on the same host, a more common configuration would be IP Availability Manager and IP Performance Manager running as a single process.

After building a data model of the discovered network topology, IP Availability Manager, IP Performance Manager, and IP Server Performance Manager use SNMP polling to monitor the topology. The results of this polling, in conjunction with the received SNMP traps, are used by:

- IP Availability Manager to diagnose the failed network objects that interrupt network connectivity.
- IP Performance Manager and IP Server Performance Manager to diagnose emerging problems in the service delivery infrastructure.

IP Availability Manager, IP Performance Manager, and IP Server Performance Manager export their topology data and analysis results (events) to the EMC Smarts Service Assurance Manager (Global Manager).

Note: IP Discovery Manager can also interface with the Global Manager.

Whenever discovery completes, the IP Manager prints a report to its log file identifying the number of network objects created during the discovery. The log file is named <Domain Manager name>.log (for example, INCHARGE-AM) and is located in the BASEDIR/smarts/local/logs directory in the IP Management Suite installation area.

IP Availability Manager Light Discovery

In addition to performing its traditional discovery of network objects, IP Availability Manager can be configured to create customized topology collection sets from the discovered network objects, one set for each of the satellite Domain Managers supported by IP Availability Manager. The term satellite, as used here refers to any of the EMC Smarts Domain Managers (such as EMC Smarts MPLS Manager™, EMC Smarts Multicast Manager™, and EMC Smarts Network Protocol Manager™) that import topology and topology updates from IP Availability Manager. (See Figure 1-2.)

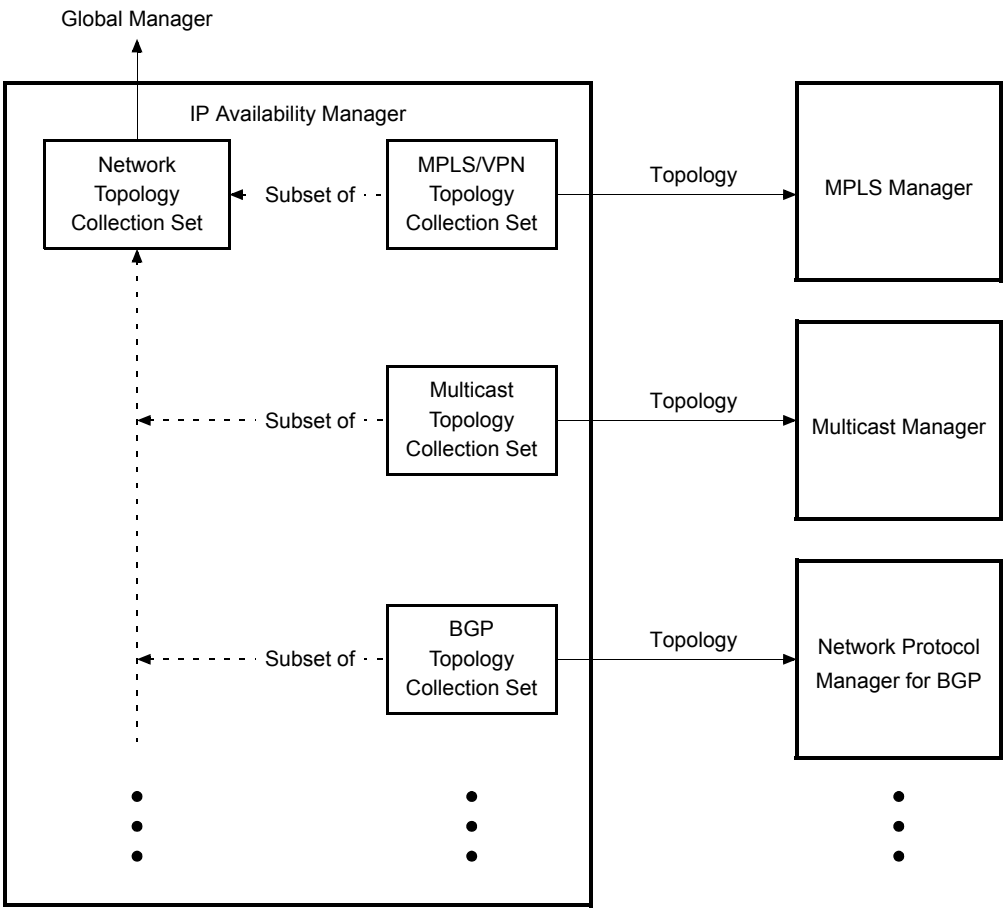


Figure 1-2 Exporting customized topology collection sets to satellite Domain Managers

From IP Availability Manager, a satellite Domain Manager not only receives its initial topology and subsequent topology updates, but it also receives Layer 1 (physical) and Layer 2 (data-link) events relevant to the imported topology. As an example of the types of objects imported from IP Availability Manager, MPLS Manager imports router objects along with the following objects associated with the routers:

- Chassis
- Card
- Port
- Interface
- IP
- DuplicateIP
- IPNetwork
- VPNIP
- VPNIPNetwork
- NetworkConnection
- SNMPAgent
- VLAN
- Partition

Appendix A, "ICIM Network Concepts," provides descriptions of these object types, and Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* provides instructions on configuring IP Availability Manager to create and export customized topology sets.

IP Availability Manager Extension for IPv6 Discovery

A feature license controls whether IP Availability Manager Extension for IPv6 software is available for configuration. (The *EMC Smarts System Administration Guide* provides information on licensing.) With this extension, IP Availability Manager can discover and monitor IPv6 addresses and diagnose IPv6 connectivity problems.

IP Availability Manager Extension for IPv6 discovers the following additional types of objects: IPv6 endpoints and IPv6 networks. It models these objects as instances of the IPv6 and IPv6Network classes, respectively. Appendix A, "ICIM Network Concepts," provides descriptions of these object types.

The *EMC Smarts IP Availability Manager Extension for IPv6 User's Guide* provides information about IP Availability Manager Extension for IPv6.

IP Availability Manager Extension for NAS Discovery

A feature license controls whether IP Availability Manager Extension for NAS software is available for configuration. (The *EMC Smarts System Administration Guide* provides information on licensing.) With this extension, IP Availability Manager can discover and monitor EMC Celerra® NAS gateways and diagnose network-attached storage (NAS) connectivity problems.

IP Availability Manager Extension for NAS discovers the following additional types of objects: Celerra NAS chassis, Celerra Control Stations, and Celerra Data Movers. It models these objects as instances of the NASChassis, Host, and FileServer classes, respectively. Appendix A, "ICIM Network Concepts," provides descriptions of these object types.

The *EMC Smarts IP Availability Manager Extension for NAS User's Guide* and the *EMC Smarts Storage Insight Management Suite Discovery Guide* provide information about IP Availability Manager Extension for NAS.

Discovery and IP tagging

By default, the IP Manager does not discover overlapping IP addresses, which are IP addresses that are reused and configured across multiple network systems (devices) in the same managed network environment. Overlapping IP addresses are prominent in an MPLS VPN environment, and common in a large global network, where private IP networks of customers often deploy identically numbered IP address spaces.

Figure 1-3 is an example of an overlapping IP address in an MPLS VPN network.

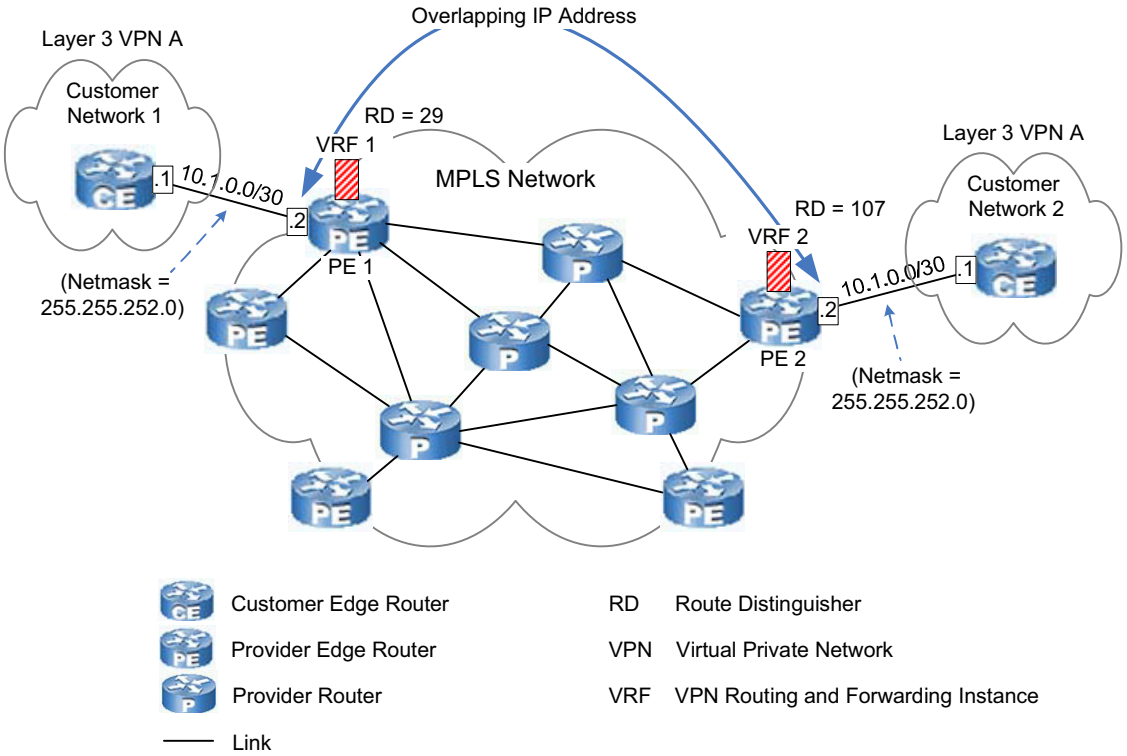


Figure 1-3 Example of an overlapping IP in an MPLS VPN network

In the example, the default behavior is as follows: The IP Manager does not create IP instances for overlapping IP address 10.1.0.2 when discovering routers PE 1 and PE 2 because, by default, the IP Manager does not discover IP addresses belonging to VPN Routing and Forwarding (VRF) tables. The VRF is a component of MPLS VPNs.

Turning on VRF IP tagging

Users can change this behavior by turning on the VRF IP tagging feature. This feature is controlled by the following two configuration parameters, both of which are in the BASEDIR/smarts/conf/discovery/tpmgr-param.conf file in the IP Manager installation directory:

- DisableOverlappingIP (TRUE by default)

  Controls whether the IP Manager discovers VRF IP addresses in the managed network environment. By default, the IP Manager does not discover VRF IP addresses.

- UseVPNIP (TRUE by default)

  Relevant only when DisableOverlappingIP is set to FALSE: Controls whether the IP Manager represents discovered VRF IP addresses as instances of the VPNIP class (the default) or instances of the IP class.

Note: VRF IPs, also known as VPN-IPs, are not discovered by default because the majority of service providers managing MPLS Layer 3 VPNs do not have the overlapping IP address condition.

Referring back to Figure 1-3, and assuming DisableOverlappingIP = FALSE and UseVPNIP = TRUE, the IP Manager will behave as follows when it encounters overlapping IP address 10.1.0.2 during discovery:

- Create two VPNIP objects for IP address 10.1.0.2.
- Tag the name of each VPNIP object with the route distinguisher value associated with the VRF to which the IP address belongs.

  The naming format is IP-<IP address>/<RD tag>. So, in this example, the IP Manager will name the two VPNIP objects as follows:

  IP-10.1.0.2/29
  (Display name: *29:10.1.0.2 [PE 1]*)

  IP-10.1.0.2/107
  (Display name: *107:10.1.0.2 [PE 2]*)

- Create a VPNIPNetwork object for each of the two VPNIP objects.
- Tag the name of each VPNIPNetwork in accordance to the following naming format: IPNET-<IP network>/<RD tag>. So, in this example, the IP Manager will name the VPNIPNetwork objects as follows:

  IPNET-10.1.0.0/29
  (Display name: *29:10.1.0.0*)

  IPNET-10.1.0.0/107
  (Display name: *107:10.1.0.0*)

For DisableOverlappingIP = FALSE and UseVPNIP = FALSE, the IP Manager will behave the same way as just described, but instead of creating two VPNIP objects and two VPNIPNetwork objects, it will create two IP objects and two IPNetwork objects.

Note: When an IP address is discovered as a VRF IP (and assuming DisableOverlappingIP is set to FALSE), the IP Manager will tag the name of the associated VPNIP or IP object with a route distinguisher value regardless of whether it is—or is not—an overlapping IP address.

The purpose of the UseVPNIP parameter is to make the current version of the IP Manager compatible with earlier versions of the satellite Domain Managers (MPLS Manager, Network Protocol Manager) that are expecting VPNIP and VPNIPNetwork objects in the topology collection set imported from IP Availability Manager. Eventually, those satellite Domain Managers will be upgraded to import VRF IP addresses and associated VRF IP networks as tagged IP and IPNetwork objects.

Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the UseVPNIP and DisableOverlappingIP parameters in the tpmgr-param.conf file and provides instructions on modifying the parameters.

Using IP tagging

Through an IP tagging feature, the IP Manager is able to discover and create IP objects for both non-VRF overlapping IP addresses and VRF IP addresses, where a non-VRF overlapping IP address is defined as an overlapping IP address that is not part of an MPLS Layer 3 VPN. In addition, users can configure this feature to double-tag the name of the objects created for discovered VRF IP addresses, or to replace the route distinguisher tag in the name of these objects with a user-specified tag. Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the IP tagging feature and provides instructions on creating IP tag filter groups.

Referring back to Figure 1-3, and assuming DisableOverlappingIP = FALSE, UseVPNIP = FALSE, user-specified tag for PE 1 = cust1, and user-specified tag for PE 2 = cust2, the IP Manager will behave as follows when it encounters overlapping IP address 10.1.0.2 during discovery:

- Create two IP objects for IP address 10.1.0.2.
- Tag the name of each IP object with (1) the user-specified tag for the IP tag filter group matching the host PE of the IP address and (2) the route distinguisher value associated with the VRF to which the IP address belongs.

  The naming format is IP-<IP address>/<user-specified tag>-<RD tag>. So, in this example, the IP Manager will name the two IP objects as follows:

  IP-10.1.0.2/cust1-29
  (Display name: *cust1-29:10.1.0.2 [PE 1]*)

  IP-10.1.0.2/cust2-107
  (Display name: *cust2-107:10.1.0.2 [PE 2]*)
- Create an IPNetwork object for each of the two IP objects.
- Tag the name of each IPNetwork object in accordance to the following naming format: IPNET-<IP network>/<user-specified tag>-<RD tag>. So, in this example, the IP Manager will name the IPNetwork objects as follows:

  IPNET-10.1.0.0/cust1-29
  (Display name: *cust1-29:10.1.0.0*)

  IPNET-10.1.0.0/cust2-107
  (Display name: *cust2-107:10.1.0.0*)

Of course, for satellite Domain Managers to take advantage of double-tagging, they must understand double-tagging. Eventually, the satellite Domain Managers (MPLS Manager, Network Protocol Manager) will be upgraded to understand double-tagging of IP and IP network objects associated with discovered VRF IP addresses.

Furthermore, all satellite Domain Managers will eventually be upgraded to understand single-tagging of IP and IP network objects associated with discovered non-VRF IP addresses. Until then, however, by default the IP Manager will create a shadow IP object for each tagged, non-overlapping, non-VRF IP object that it creates, and create a shadow IPNetwork object for the IPNetwork object to which the IP object is "ConnectedVia." The names of the shadow objects are not tagged except for their display names, which are prepended with "[Shadow]." Earlier versions of satellite Domain Managers will import the shadow objects, rather than their tagged counterparts, when importing topology from IP Availability Manager.

Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* provides more information about shadow IP and shadow IPNetwork objects.

IPSec discovery

The IP Manager discovers and monitors IPSec tunnels and Internet Key Exchange (IKE) tunnels between IPSec-enabled routers. IPSec is a standardized framework for securing IP communications at the network layer, and IKE is a standardized key exchange protocol used to set up IPSec tunnels.

The IP Manager reads the Routing Table of IPSec-enabled routers to discover the terminating endpoints (subinterfaces) of IPSec tunnels and IKE tunnels. It layers the subinterfaces over the associated physical interfaces on the local and remote IPSec-enabled routers, and creates network connections between the local and remote subinterfaces to model the IPSec and IKE tunnels.

The IP Manager uses Router, Interface, IP, and NetworkConnection object types to model IPSec and IKE tunnels. It monitors the status of the tunnels by monitoring the status of the underlying physical interfaces.

IPSec and IKE tunnels are dynamic in the sense that they can move to other interfaces when an interface fails. Accordingly, when an underlying interface fails, the IP Manager rediscovers the hosting router for the purpose of rediscovering the IPSec and IKE tunnels. For the rediscovery to occur, the autoReprobe_short configuration parameter in the BASEDIR/smarts/conf/discovery/discovery.conf file must be set to TRUE.

Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in the discovery.conf file and provides instructions on modifying the parameters. "Short discovery for IPSec tunnels" on page 2-20 describes the short discovery process.

Load balancer discovery

The IP Manager discovers and monitors load balancers, which are servers that distribute client requests across multiple web servers. A load balancer presents a single virtual server to the client and transparently routes the client requests to the web servers.

Figure 1-4 is an example of an a load balancer deployment having one virtual IP and multiple servers.

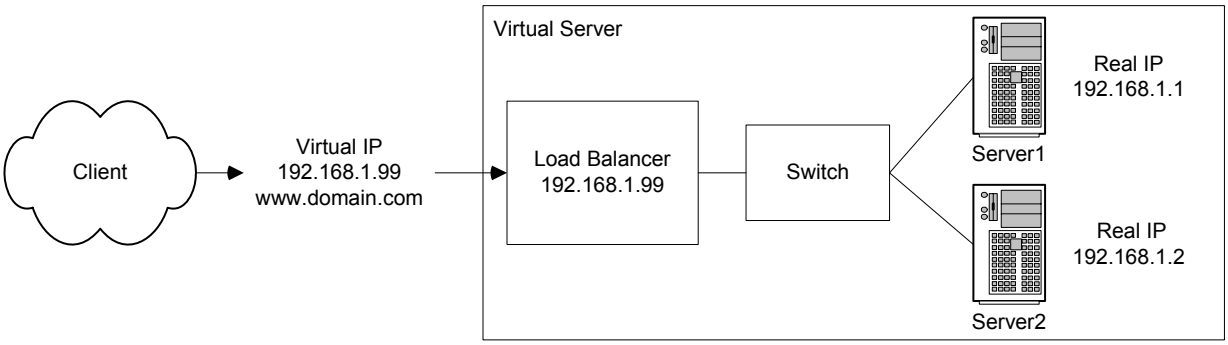


Figure 1-4 Example of a load balancer deployment

In the example, all traffic for virtual IP 192.168.1.99 is load balanced across real servers with IP 192.168.1.1 and 192.168.1.2. The virtual server appears in the Domain Name System (DNS) as a single, indivisible entity.

The IP Manager discovers the host running the load balancer, the virtual IPs hosted by the load balancers, and the virtual-to-real IP mappings for each virtual IP. It uses LoadBalancer and VirtualTCP object types to model the load balancers.

The IP Manager monitors the status of the load balancers by monitoring the status of the load balancer host.

2

Discovery Process

This chapter identifies the methods of discovery for the IP Manager and explains the phases of the discovery process. It consists of the following sections:

- Discovery methods 2-2
- Discovery phases 2-5
- Autodiscovery data sources 2-19
- Short discovery for IPSec tunnels 2-20

Discovery methods

Figure 2-1 and Figure 2-2 show the discovery methods and process for adding systems (devices) to the topology modeled by the IP Manager. The discovery methods are:

- Manual discovery
- Autodiscovery
- Topology import

Manual discovery

With manual discovery (Figure 2-1), the IP Manager discovers a set of systems specified in a seed file, or discovers an individual system specified in an Add Agent command. This method is useful in networks where the topology information is available from a well-maintained database.

Using this method with a comprehensive seed file is the quickest method for discovering a network because all systems are discovered using the IP addresses or system names that are already in the seed file. Chapter 6, "Preparing a Seed File," provides instructions on creating a seed file.

Autodiscovery

With autodiscovery (Figure 2-1), the IP Manager automatically discovers candidate systems based on the systems initially added to the modeled topology using manual discovery. This method is useful when topology information is incomplete or unavailable or when manual updates cannot be maintained

When enabled, autodiscovery takes IP addresses obtained from discovered systems to find additional discovery candidate systems. The initial IP addresses available to autodiscovery correspond to the neighboring systems of the systems discovered using manual discovery.

Autodiscovery requires discovery filters, described in Chapter 7, "Creating Discovery Filters." For systems that match a discovery filter and are added to the modeled topology, they too are probed for IP addresses of their neighbors. The autodiscovery cycle continues until no more new IP addresses match the discovery filters. The discovery filters, which are configurable, ensure that the IP Manager sweeps only the networks that the customer systems are configured to access.

Topology import

With topology import (Figure 2-2), the IP Manager discovers a set of candidate systems imported from a third-party source through an EMC Smarts IP adapter. The IP adapters are described in the *EMC Smarts IP Adapters User's Guide*.

Topology import, as mentioned in Chapter 7, "Creating Discovery Filters," requires at least one discovery filter. The filter should be defined broadly enough to accept the possible IP addresses and system names provided by the third-party software.

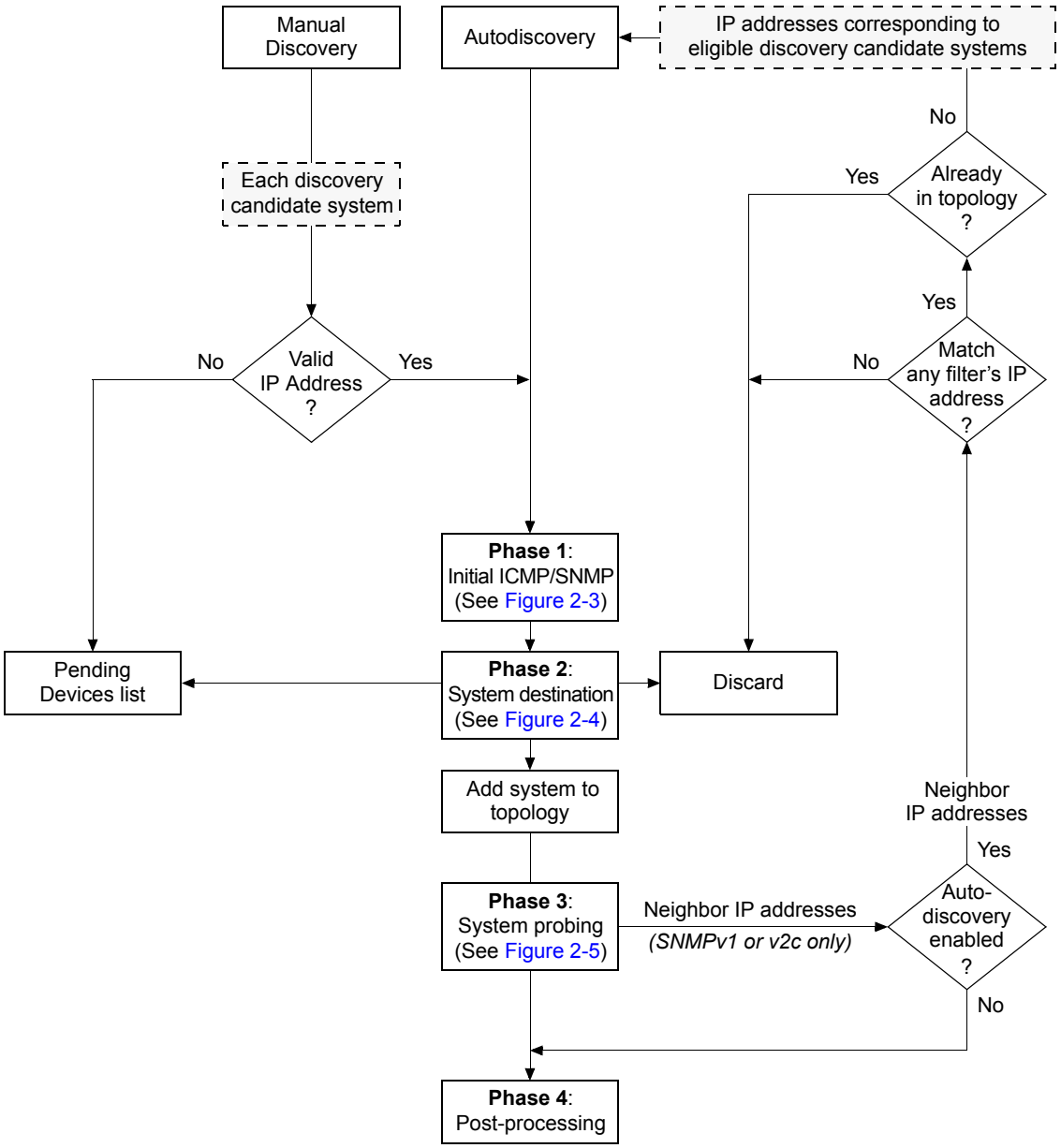


Figure 2-1 Discovery process using manual discovery and autodiscovery

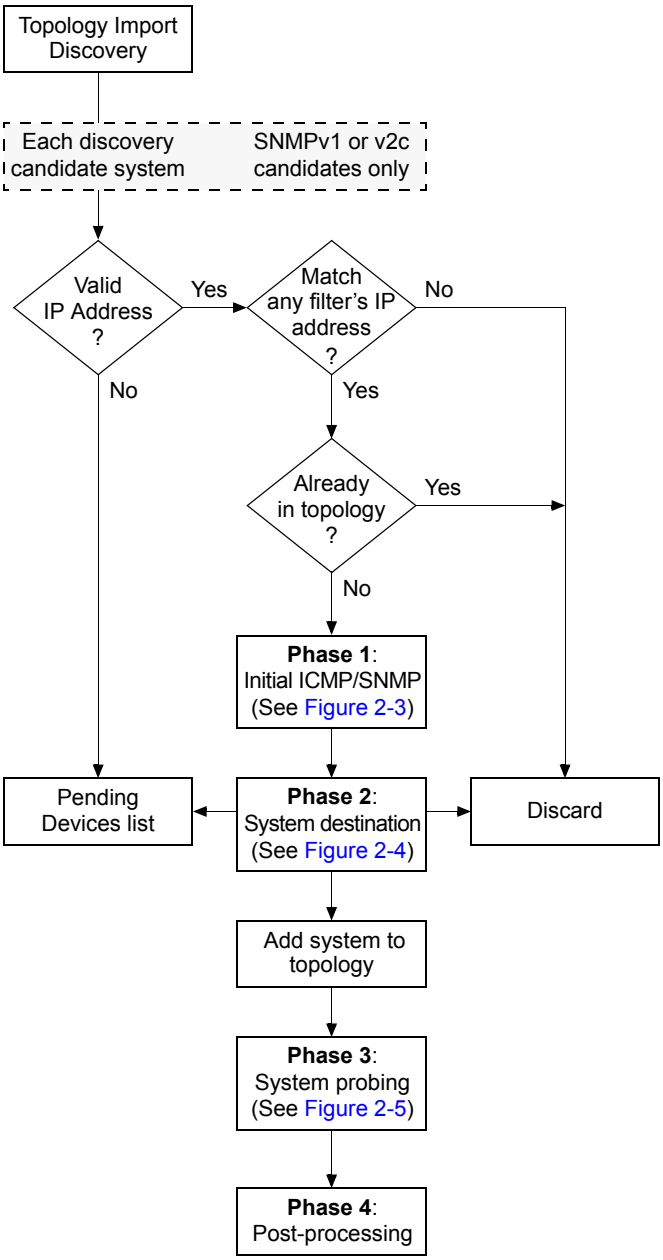


Figure 2-2 Discovery process using topology import

Discovery phases

During discovery, the IP Manager uses certain object classes to create representations in its repository of the managed network systems, their dependencies, their internal components, and their connections. The topology representations created by the IP Manager are identified and described in Appendix A, "ICIM Network Concepts."

The discovery is divided into four phases, shown in Figure 2-1 and Figure 2-2, that are performed sequentially for each discovery candidate system:

1. Perform initial polling of the candidate system
2. Determine the destination of the candidate system: add to topology, place on Pending Devices list, or discard
3. Probe the system added to the topology
4. Post-process the discovery information

Phase 1: Perform initial polling of the candidate system

Discovery for each discovery candidate system begins with a hostname or an IP address, provided by the user or automatically discovered. The discovery process resolves the hostname to one or more IP addresses through the local name resolution service.

The first phase of the discovery process, shown in Figure 2-3, determines if the discovery candidate system is reachable and sends an initial SNMP request to identify the system.

The discovery process pings the IP address of the discovery candidate system to see if the address is a reachable address. If the address is reachable, the discovery process sends an SNMP request to the system requesting the following system-related information:

- sysName
- sysDescr
- sysObjectID
- sysContact
- sysLocation

By default, the discovery process sends an SNMPv2c request to the candidate system; if that request fails after the allotted number of retries, the discovery process sends an SNMPv1 request. If the SNMP version is explicitly specified (V1, V2C, V3) for the candidate system (via a discovery filter, a seed file, or an Add Agent command), the discovery process only sends an SNMP request using the specified version.

Note: Discovery filters support SNMP version V1 and V2C but not V3.

For an SNMPv1 or v2c request to the candidate system having multiple read community strings, the discovery process sends multiple SNMP requests to the system simultaneously, each containing a different community string. As described in "Adding and removing read community strings" on page 11-2, you can specify multiple read community strings for a candidate system.

The results of the ICMP and SNMP polling serve as input to the second phase of the discovery process.

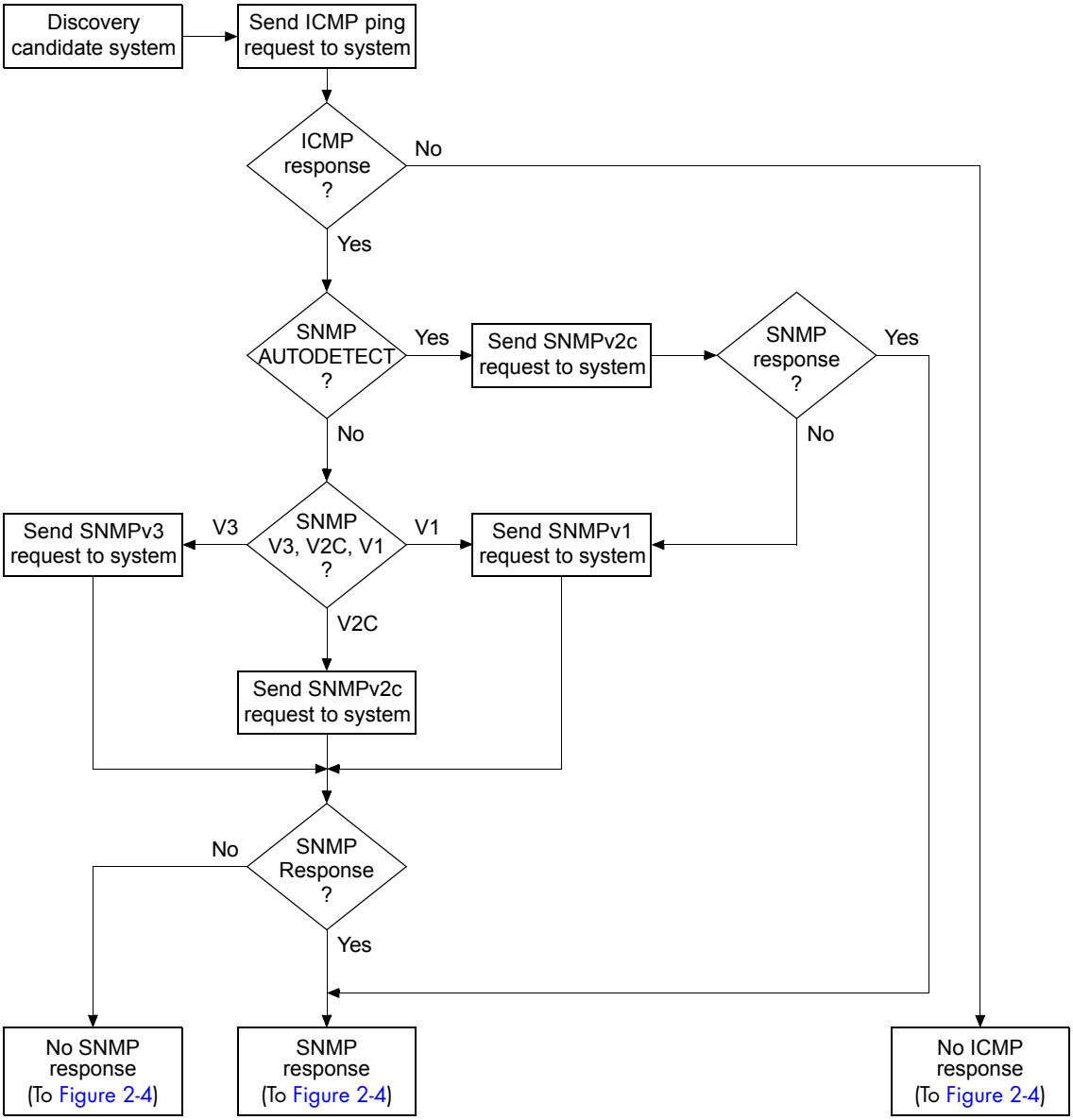


Figure 2-3 Phase 1: Initial polling of a candidate system

Phase 2: Determine the destination of the candidate system

The second phase of the discovery process, shown in Figure 2-4, determines the destination of a discovery candidate system: add to topology, place on the Pending Devices list, or discard.

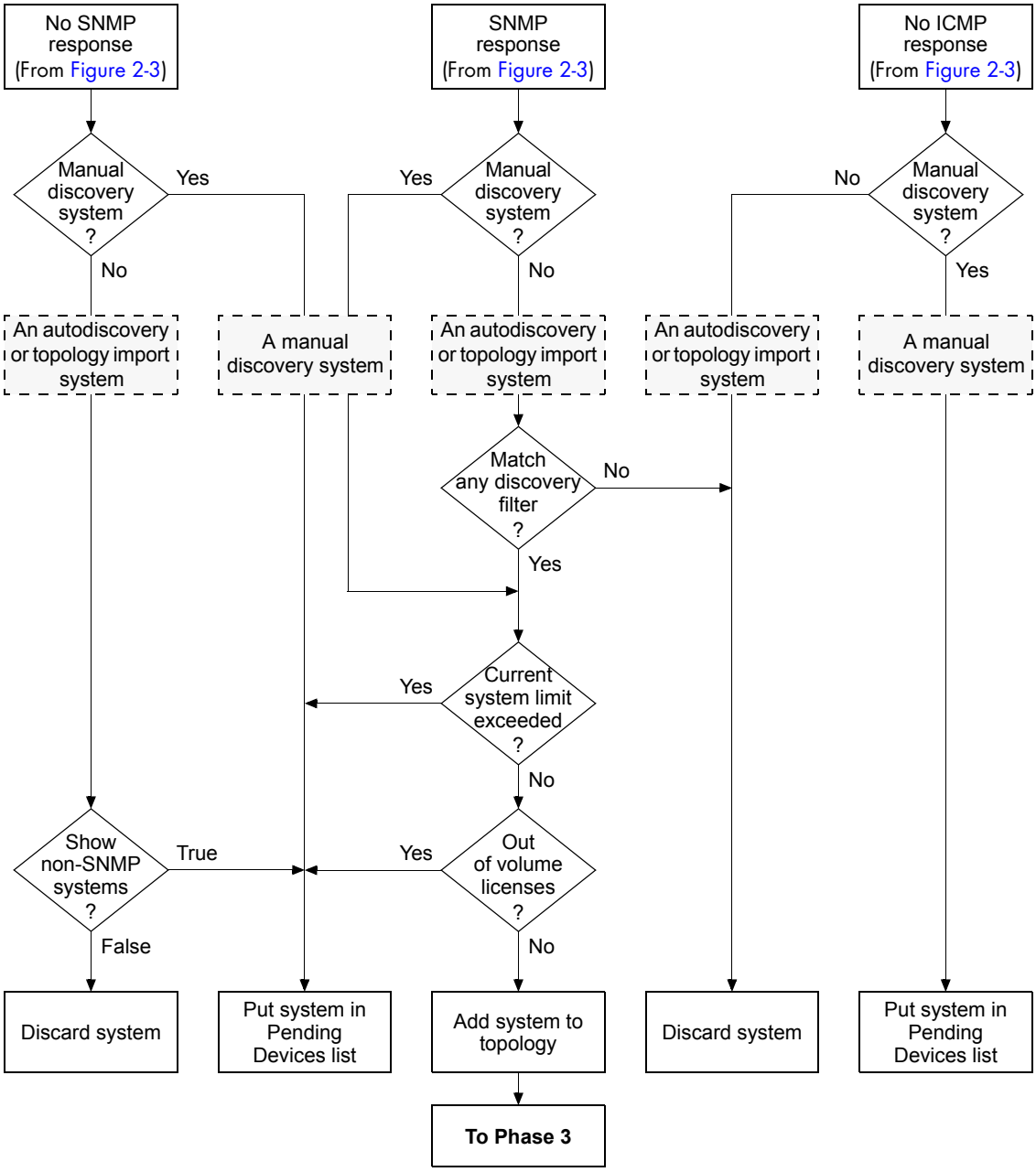


Figure 2-4 Phase 2: Determining the destination of a candidate system

As shown in Figure 2-4, the destination of the candidate system depends on many factors, including:

- Whether the candidate system is an autodiscovery, manual, or topology import candidate system
- Whether the system passes ICMP/SNMP polling
- Whether the system matches a discovery filter
- Whether the current system limit is exceeded

- Whether any volume licenses are available
- Whether ShowPendingNONSNMP in the BASEDIR/smarts/conf/discovery/discovery.conf file is set to TRUE

Table 2-1 summarizes how these factors determine the destination of a candidate system when sufficient volume licenses are available. "Insufficient number of volume licenses" on page 8-7 provides information about volume licenses and the OutOfLicense notification.

Table 2-1 Factors affecting the destination of a candidate system

Candidate system	Initial polling		Match filter	Current system limit exceeded*	ShowPendingNONSNMP	Destination
	ICMP	SNMP				
Autodiscovery or topology import	Passed	Passed	Yes	No	N/A	Topology
				Yes	N/A	Pending Devices list
			No	N/A	N/A	Discard
		Failed	N/A	N/A	TRUE	Pending Devices list
					FALSE (default)	Discard
	Failed	N/A	N/A	N/A	N/A	Discard
Manual Discovery	Passed	Passed	N/A	No	N/A	Topology
				Yes	N/A	Pending Devices list
		Failed	N/A	N/A	N/A	Pending Devices list
	Failed	N/A	N/A	N/A	N/A	Pending Devices list

* 50 by default; to change the default, see "Setting a current system limit for the topology" on page 4-4.

The "Match filter" column in the table pertains to the matching of discovery-filter criteria other than IP address. An autodiscovery candidate system is subjected to two levels of discovery-filter matching: initially, a matching of IP address, and then a matching of other criteria after the candidate system passes ICMP/SNMP polling. The two levels of discovery-filter matching are described in Chapter 7, "Creating Discovery Filters."

If a candidate system cannot be discovered and is not discarded, the IP Manager places the system on the Pending Devices list. The Pending Devices list is described in "Pending Devices list" on page 8-12.

Phase 3: Probe the system added to the topology

During the third phase of the discovery process, shown in Figure 2-5, each system destined for the topology is probed to gather additional information about the system, including the system's components (such as cards, ports, interfaces, and so on), IP addresses, VLAN membership, environment sensors, system resources (such as memory, CPU, and so on), IP networks, netmasks, and neighboring systems. The IP Manager uses this information to create the topology for the system.

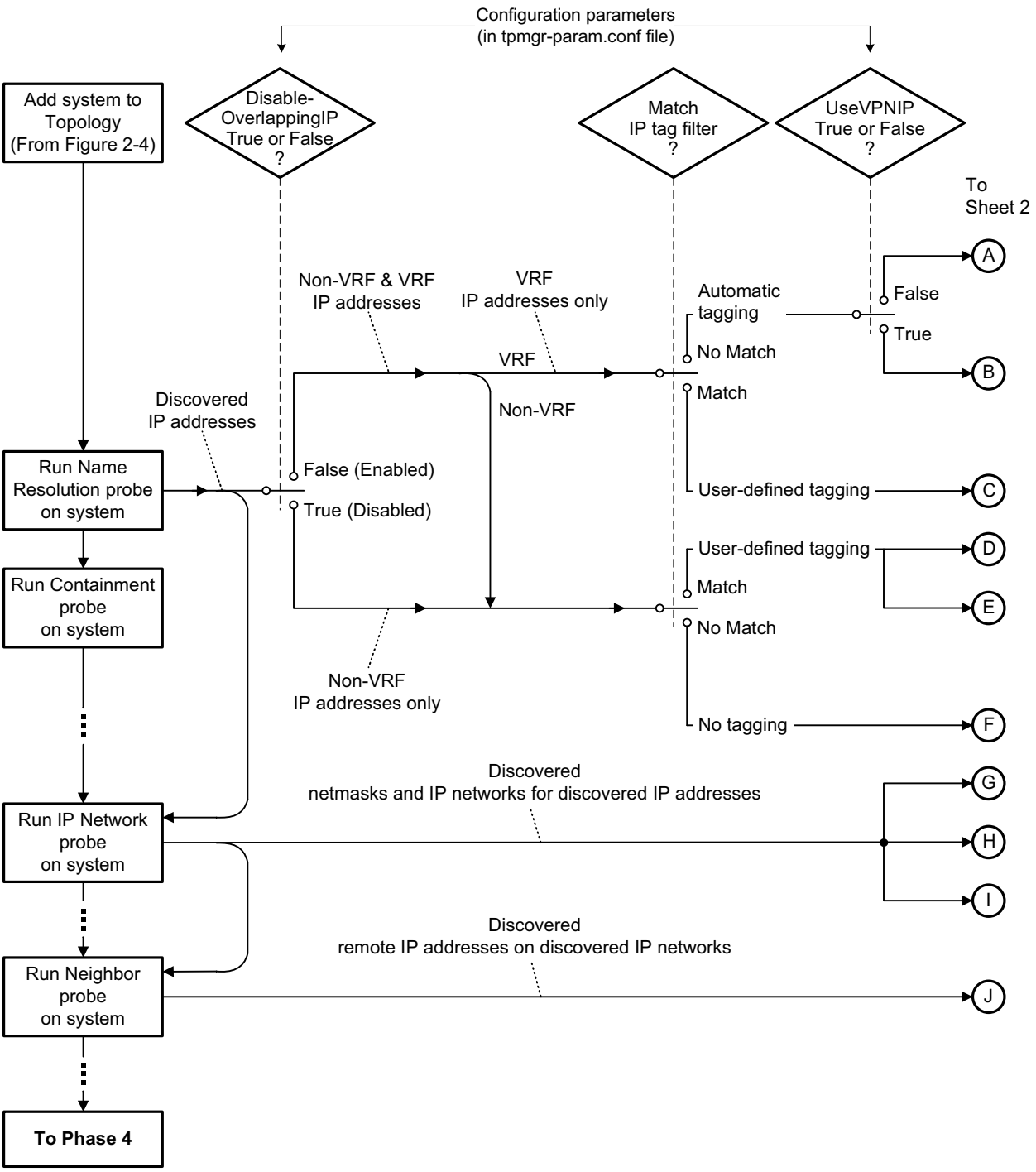


Figure 2-5 Phase 3: Probing the system added to the topology (Sheet 1 of 2)

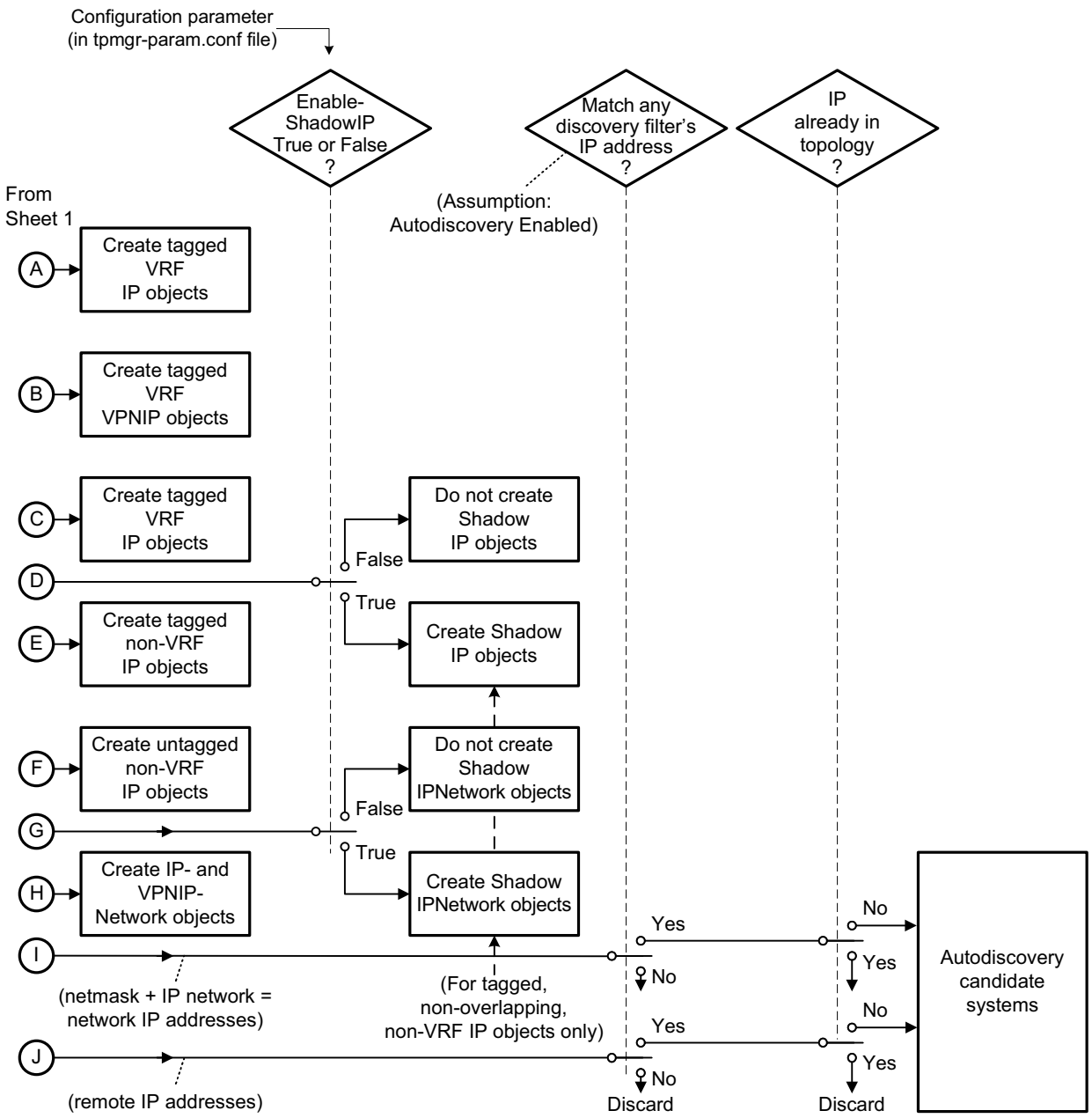


Figure 2-5 **Phase 3: Probing the system added to the topology (Sheet 2 of 2)**

Among other things, the network IP addresses formed from the discovered IP networks and netmasks serve as input to the autodiscovery process. Autodiscovery, if enabled, will compare each of these IP addresses against one or more discovery filters and add each IP address matching a filter to the discovery queue. Discovery filters are described in Chapter 7, "Creating Discovery Filters."

Note: Autodiscovery is not supported for systems having SNMPv3 agents.

The discovery process uniquely identifies the system within the topology and determines the system's level of certification. These tasks are accomplished using the sysName, sysDescr, and sysObjectID information retrieved from the initial SNMP poll.

oid2type configuration files

The discovery process matches the sysObjectID to an OID in the oid2type configuration files to identify the system's type (Switch, Router, Hub, Bridge, Host), its vendor, its model, its certification level, what discovery probes to use to probe the system, and what instrumentation classes to use to monitor the system. The discovery process searches for the probes in the .import files, which are located in the same directory as the oid2type files: BASEDIR/smarts/conf/discovery.

Certification

Certification is a process by which EMC Corporation ensures that the devices (specific models of systems) managed by the IP Manager are properly represented in the modeled topology and that the devices are correctly monitored for connectivity and performance analysis. There are three major parts to the certification process:

- Discovery of the device, including the device's components (such as cards, ports) and access points (such as IP addresses, MAC addresses).
- Building of Layer 2 (data-link) and Layer 3 (network) connectivity.
- Verification of operational status of the device and its components.

The *EMC Smarts 7.0 Certified Device List* identifies the vendor-specific models supported by the IP Manager.

Certification levels

A device may have one of three certification levels:

- CERTIFIED —EMC Corporation has successfully tested a simulation of the device using standard and proprietary MIBs supplied by the vendor.

  The IP Manager discovers the device using standard MIB-II data and proprietary MIBs.

- TEMPLATE— EMC Corporation assumes that the device supports some standard and proprietary MIBs but has not tested the device.

  The IP Manager recognizes the OID but has no information about the MIBs that this device supports.

- GENERIC — EMC Corporation does not have the OID for the device.

  The IP Manager does not recognize the OID and responds by classifying the certification level of the device as GENERIC, adding the device to the Node class, and analyzing the device using MIB-II data.

"Certifying non-network systems" on page 9-2 describes how to add support for new devices.

Standard and proprietary MIBs

The IP Manager uses proprietary MIBs (also known as enterprise, vendor-specific, or private MIBs), in addition to standard MIBs (also known as generic MIBs), because standard MIBs do not always contain the information needed by the IP Manager to discover and manage certain objects. Standard MIBs, for example, do not represent the processor or memory objects necessary for performance analysis. In these cases, proprietary MIBs offer more detailed information.

Discovery probes

The probing of topology is accomplished using the discovery probes (drivers) identified in the following .import files:

```
BASEDIR/smarts/conf/discovery/DISCOVERY_<model>.import

BASEDIR/smarts/conf/discovery/DISCOVERY_CUSTOM.import
```

The main probes are identified in the DISCOVERY_<model>.import files, and the custom probes, also known as discovery hook scripts, are identified in the DISCOVERY_CUSTOM.import files. "Using discovery hook scripts" on page 9-4 provides information about the custom probes.

Note: By default, the custom probes are not configured and therefore do not run.

Table 2-2 identifies the discovery probes available to the IP Manager. The IP Manager runs the discovery probes in the order shown.

Table 2-2 Discovery probes

Probe name	Applicable to which systems?	Available to which IP Managers?
Name Resolution	All systems	All IP Managers
Containment	All systems	All IP Managers
VLAN	Bridges, switches	All IP Managers
Bridge	Bridges, switches	All IP Managers
HSRP	Routers	All IP Managers**
IPV6	All systems	IP Availability Manager Extension for IPv6***
IP Network	All systems	All IP Managers
Virtual Router	Routers, switches	All IP Managers
Neighbor	All systems	All IP Managers
Performance*	All systems	IP Availability Manager and IP Performance Manager running as a single process, IP Performance Manager, IP Server Performance Manager, IP Discovery Manager, IP Discovery Manager and IP Performance Manager running as a single process
Host Resources*	All systems	IP Server Performance Manager****

* Performance probes and Host Resources probes are also known as Health probes.
** Requires a feature license to enable the HSRP probe.
*** Requires a feature license to enable the IPV6 probe; this probe is also available to all other IP Managers when the license is installed.
**** Requires a feature license to enable the Host Resources probe.

Note: IP Availability Manager Extension for NAS requires a feature license to enable additional Containment probes, which discover Celerra NAS components. These additional Containment probes are also available to all other IP Managers when the license is installed.

The actual probes that run for a specific system are specified in the oid2type configuration files. (Most of the probes are vendor-specific.) If a probe is not specified for a system, the IP Manager skips the probe.

The probes operate in series, one right after the other. In the event that one of the discovery probes is unable to complete, the discovery process does not stop. Instead, discovery proceeds with the next probe. Events that can prevent a discovery probe from completing are discussed in Chapter 8, "Understanding Discovery Results."

Name Resolution probe

The Name Resolution probe consists of multiple probes, including:

- A probe that gathers IP addresses and other MIB-II data for the system
- A probe that gathers VRF IP addresses and route distinguisher tags from individual vendor MIBs on the system.
- A probe that provides a unique name for the system—described in "How discovery names a system" on page 3-4

The Name Resolution probe discovers almost all of the IP addresses that are available for the system, including the VRF IP addresses if the DisableOverlappingIP configuration parameter is set to FALSE. Table 2-3 summarizes the IP-related objects created by the Name Resolution probe.

Table 2-3 Matrix of IP-related objects created by the Name Resolution probe

Configuration parameters in tpmgr-param.conf file*			VRF IP addresses discovered?	VPNIP or IP objects created?	Shadow IP objects created?
DisableOverlappingIP	UseVPNIP	EnableShadowIP			
TRUE (default)	Don't care	Don't care	No	Not applicable	Not applicable
FALSE	TRUE (default)	Don't care	Yes	VPNIP	No
FALSE	FALSE	Don't care	Yes	IP	No
Don't care	Don't care	TRUE (default)	Not applicable	IP	Yes**
Don't care	Don't care	FALSE	Not applicable	IP	No

* First three rows relate to VRF IP addresses, and last two rows relate to non-VRF IP addresses.
** Shadow IP objects are created only for tagged, non-overlapping, non-VRF IP objects.

Note: Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* provides more information about shadow IP objects.

All VPNIP objects created by the Name Resolution probe are automatically tagged: The name of a VPNIP object is tagged with the route distinguisher value of the VRF to which the IP address belongs. VPNIP objects were introduced in an earlier release of the IP Manager as a work-around to solve the issue of duplicate IP objects in customer networks that support MPLS VPNs.

All VRF and non-VRF IP objects created by the Name Resolution probe are subject to user-defined IP tagging. (Assumes that DisableOverlappingIP is set to FALSE.) Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* provides instructions on creating IP tag filter groups and explains the naming scheme for applying tags to IP objects.

When the EnableShadowIP parameter is set to TRUE (default), the Name Resolution probe creates a shadow IP object for each tagged, non-overlapping, non-VRF IP object. The probe uses the following naming scheme to name the shadow IP object:

IP-<IP address>
(Display name: *[Shadow]:<IP address> [<system containing this IP>]*)

Where <*IP address*> is the IP address of the IP object that the shadow IP object is shadowing.

Note: A shadow IP object has an IsShadow attribute set to TRUE to identify it as a shadow IP object. It also has a ShadowOf relationship with the IP object that it is shadowing.

All IP address and MIB information gathered by the Name Resolution probe for the discovered system is made available to all other discovery probes, including the IP Network probe.

Containment probe

The Containment probe discovers the components of a system including ports, interfaces, MAC endpoints, and any cards or modules. It queries the following MIB variables:

- ifType
- ifDescr
- ifSpeed
- ifMtu
- ifPhysAddress
- ifAlias
- ifName
- Other variables

If the Containment probe has access to vendor-specific MIBs, it queries additional variables to gather additional information, such as:

- Duplex mode (full-duplex, half-duplex) of ports and interfaces
- Data Link Connection Identifiers (DLCIs) for Frame Relay virtual connections
- IPSec and Internet Key Exchange (IKE) interfaces for IPSec and IKE tunnels
- Virtual and real IP address and port information for load balancer virtual connections

For bridges or switches, the Containment probe might also query the dot1dBasePortIfIndex MIB variable.

VLAN probe

The VLAN probe collects VLAN information for the system (bridge, switch), including VLAN identifiers, VLAN trunks, VLAN trunk ports, and VLAN port memberships. It uses vendor-specific MIB information.

The VLAN probe creates VLAN objects and VLAN members (Port or Interface objects), and uses the PartOf/ComposedOf relationship to associate the VLAN objects with their VLAN members.

All VLAN objects created by the VLAN probe are subject to user-defined VLAN tagging, which ensures that bridges and switches are assigned to the appropriate VLAN. Chapter 7, "Customizing Polled Data Thresholds," in the *EMC Smarts IP Management Suite Configuration Guide* provides instructions on creating VLAN tag groups and explains the naming scheme for applying tags to VLAN objects.

Bridge probe

The Bridge probe collects bridge-forwarding tables and Spanning Tree Protocol (STP) information for the system (bridge, switch). Discovery uses this information to create MAC objects and to build the bridging relationships between MAC endpoints and ports. The Bridge probe uses standard and vendor-specific MIB information.

One Bridge probe is run for each VLAN discovered on a bridge or switch, to gather the bridge-forwarding and STP information for the VLAN.

HSRP probe

The HSRP probe collects information from a redundancy group of routers involved in an HSRP group; the routers communicate with one another using the proprietary Cisco Hot Standby Router Protocol. The HSRP probe discovers the group number, the virtual IP address, and the virtual MAC address for an HSRP group; the priority number and status for each router (endpoint) participating in the group; and the currently active router in the group. The HSRP probe uses vendor-specific MIB information.

The HSRP probe creates HSRPGroup objects and HSRPEndpoint objects.

IPV6 probe

The IPV6 probe collects IPv6 address and netmask information for the system. The discovery process uses this information to create IPv6 endpoint and IPv6Network objects and to build the connected relationships between IPv6 endpoints and IPv6 networks. The IPV6 probe uses standard and vendor-specific MIB information.

The IPV6 probe uses IPv4 SNMP polling to learn about the IPv6 addresses and networks. After the IPv6 and IPv6Network objects are created, the IP Manager uses IPv6 ICMP polling to monitor the IPv6 objects.

Note: Currently, the IP Manager supports IPv4 SNMP, IPv4 ICMP, and IPv6 ICMP polling, but does not support IPv6 SNMP polling.

IP Network probe

The IP Network probe examines the system's IP address table (ipAddrTable) to combine the IP addresses and the netmask for each interface on the system to create VPNIPNetwork and IPNetwork objects, and to build the connected relationships between IP endpoints and IP networks. The IP Network probe uses standard MIB-II information.

All VPNIPNetwork objects created by the IP Network probe are automatically tagged: The name of a VPNIPNetwork object is tagged with the route distinguisher value assigned to the VPNIP objects "ConnectedVia" the VPNIPNetwork object. VPNIP and VPNIPNetwork objects were introduced in an earlier release of the IP Manager as a work-around to solve the issue of duplicate IP objects in customer networks that support MPLS VPNs.

All VRF and non-VRF IPNetwork objects created by the IP Network probe are subject to user-defined IP tagging. Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* provides instructions on creating IP tag filter groups and explains the naming scheme for applying tags to IP objects and IPNetwork objects.

When the EnableShadowIP parameter is set to TRUE (default), the IP Network probe creates a shadow IPNetwork object for each IPNetwork object to which a tagged, non-overlapping, non-VRF IP object is "ConnectedVia." The probe uses the following naming scheme to name the shadow IPNetwork object:

IPNET-<IP network>
(Display name: *[Shadow]:<IP network>*)

Where *<IP network>* is the IP network of the IPNetwork object that the shadow IPNetwork object is shadowing.

Note: A shadow IPNetwork object has an IsShadow attribute set to TRUE to identify it as a shadow IPNetwork object. It also has a ShadowOf relationship with the IPNetwork object that it is shadowing.

The IP Network probe also combines each discovered IP network with its netmask to form network IP addresses, for input to the autodiscovery process. (For example, consider IP network 192.168.71.0 having a netmask of 255.255.255.0: The IP Network will create network IP addresses in the range 192.168.71.<1-254> for this IP network and netmask.) Any network IP address that matches a discovery filter and is not already in the modeled topology becomes an autodiscovery candidate system.

Virtual Router probe

The Virtual Router probe discovers the virtual routers in a physical router or switch that contains virtual routers. The Virtual Router probe discovers the following additional information: the physical card on which the virtual router resides (if such information is available) and the virtual router's interfaces and their associated physical ports. The Virtual Router probe uses vendor-specific MIB information.

The Virtual Router probe creates a Router object for the physical router (or a Switch object for the physical switch), creates a Router object for each virtual router, and uses the PartOf/ComposedOf relationship to associate the virtual-router objects with the physical-router (or physical-switch) object.

Neighbor probe

The Neighbor probe collects information about a system's neighbors. It reads Cisco Discovery Protocol (CDP) tables, Extreme Discovery Protocol (EDP) tables, Foundry Discovery Protocol (FDP) tables, and other tables to discover remote addresses on the already discovered networks and to discover ATM information. The Neighbor Probe uses vendor-specific MIBs.

The Neighbor probe passes the remote IP addresses to the autodiscovery process, which adds the IP addresses to its list of potential discovery candidate systems. The probe does not create IP objects for the remote IP addresses.

The discovery process checks to see if any of the remote IP addresses correspond to IP objects already in the repository. The discovery process will use this information during the final phase of discovery to create cable, trunk cable, and network connections.

Performance probe

The Performance probe collects performance-related information for system objects such as ports, interfaces, processors, memory, fans, power supplies, voltage sensors, and temperature sensors. The Performance probe uses standard and vendor-specific MIB information.

The Performance probe creates Processor, Memory, Fan, PowerSupply, VoltageSensor, and TemperatureSensor objects and monitors statistics for the objects. For Interface or Port objects (created by the Containment probe), the performance monitoring might be the utilization based on the octets received and transmitted. For Memory objects, the performance monitoring might be the total amount of available memory, the total number of overrun buffers, and so on.

Host Resources probe

The Host Resources probe collects additional performance-related information for system objects. It collects metrics about host disks and filesystems as well as additional information about host processors and memory. The Host Resources probe uses standard and vendor-specific MIB information.

The Host Resources probe creates Disk and FileSystem objects and monitors statistics for the objects. It also monitors additional statistics for the Processor and Memory objects created by the Performance probe.

Discovered SNMP agent

In addition to discovering the system, the discovery process also discovers the system's SNMP agent and adds it to the topology. The discovery process sets the following attributes for the SNMP agent:

- Sets the AgentAddress attribute to the IP address used by the initial ICMP poll to reach the system.
- Sets the AgentAddressList attribute to a list of IP addresses discovered on the system—includes the IP address used by the initial ICMP poll to reach the system.
- Sets the PortNumber attribute (161 by default) to the port number used by the first SNMP poll to successfully communicate with the SNMP agent.
- Sets the SNMPVersion attribute (AUTODETECT by default) to the version used by the first SNMP poll to successfully communicate with the SNMP agent.
- (SNMPv1 or v2c only): Sets the ReadCommunity attribute (public by default) to the read community string used by the first SNMP poll to successfully communicate with the SNMPv1 or v2c agent.
- (SNMPv3 only): Sets the attributes shown in Table 2-4 to the values used by the first SNMP poll to successfully communicate with the SNMPv3 agent:

Table 2-4 Values used in SNMP polling

Attribute	Description
User	Name of user included in SNMP request sent to this agent.
AuthProtocol	Protocol used to authenticate SNMP request sent to this agent: MD5, SHA, or NONE (default).
PrivProtocol	Protocol used to encrypt SNMP request sent to this agent: DES, AES, or NONE (default).
EngineID	Identifier that uniquely identifies SNMP engine of this agent; for example, 800002b804616263.
EngineBoots*	Number of times that SNMP engine of this agent has initialized or reinitialized since its last configuration.
EngineBootTime*	Date and time that SNMP engine of this agent last initialized or reinitialized.
* Retrieved from SNMP polling of this agent.	

- Sets the sysName, sysDescr, sysObjectID, sysContact, and sysLocation attributes to the sysName, sysDescr, sysObjectID, sysContact, and sysLocation values retrieved from the first SNMP poll to successfully communicate with the SNMP agent.
- Sets many other SNMP agent attributes according to the values found by the discovery probes.

Discovered VR Agents

For a physical router or switch containing virtual routers, the discovery process also discovers the VR agents for the virtual routers and adds those agents to the topology. Because VRAgent is a subclass of SNMPAgent, the attributes of a VR agent are the same as the attributes of an SNMP agent with the following exception: the VR agent has an additional attribute called RouterID.

Phase 4: Post-process the discovery information

During the final phase of discovery, post-processors consolidate the information collected from the various probes to create Layer 2 (data-link) and Layer 3 (network) connectivity between the systems in the topology. The post-processors base the connectivity on IP networks, standard protocols, and vendor-specific discovery protocols.

The post-processing basic functions include:

- Mapping IP and MAC access points retrieved from the ARP Cache to the appropriate systems. For non-SNMP systems, create IP and MAC access points and layer them over a physical interface. Because non-SNMP systems are discovered, post-processors create an artificial physical interface for each IP address associated with the system.
- Removing (pruning) MAC access points that do not belong to systems in the topology. Any MAC access point bridged by a port but not hosted by any system is removed.
- Creating connections based on bridge-forwarding information: create cables between ports and interfaces, create trunk cables between ports, and remove MAC access points that are bridged by access ports from the BridgedVia relationships of other ports.
- Creating trunk cables based on STP information.
- Creating network connections to represent WAN, or logical, connections.
- Creating connections based on discovery protocols.
- Creating any user-defined connections specified in the user-defined-connections.conf file.
- Creating partitions, if any, specified in the user-defined partition.conf file.

The user-defined-connections.conf and partition.conf files are located in the BASEDIR/smarts/conf/discovery directory. Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in these files and provides instructions on modifying the parameters.

Autodiscovery data sources

The Name Resolution, IP Network, and Neighbor probes contact the autodiscovery data sources to gather the topology and IP address table (ipAddrTable) MIBs of the seed systems to determine what other systems are known to the seed systems. As these systems are discovered by autodiscovery, they too are probed for candidate systems.

The autodiscovery process uses three different sources of information to discover candidate systems:

- Topology MIB
- IP address table MIB
- ARP table MIB

By default, autodiscovery uses the topology and IP address table MIBs of the discovered systems (initially, of the discovered seed systems) because the MIBs are very reliable sources for topology information. The procedure for adding the Address Resolution Protocol (ARP) table MIB as a data source is presented in "Select autodiscovery data sources" on page 4-5.

Topology MIBs data source

Topology MIBs are an important data source for discovering neighboring systems. When the topology MIB data source is enabled, autodiscovery uses the IP addresses from these sources to discover related systems. Examples of topology MIB data sources include:

- IP addresses stored in the Cisco Discovery Protocol (CDP) tables
- IP addresses stored in the MIB-II atmInterfaceConfTable table

In some cases, however, a system is not discovered for each IP address, if, for example, there is no route to the network from the host on which the IP Manager is running. This situation can occur when security restrictions are designed to prevent access between two separate networks.

IP address table MIB data source

The MIB-II IP address table lists all of the IP interfaces of the system being discovered as well as the subnet masks associated with these IP networks. Using this information, autodiscovery derives all of the possible IP addresses within each subnet. The Name Resolution probe pings each of these addresses to find candidate systems for subsequent SNMP probing.

ARP table MIB data source

The MIB-II ARP table provides a mapping of IP addresses (Layer 3) to MAC addresses (Layer 2) for any neighboring system that has recently communicated with the system with this MIB. However, because the ARP MIB is only populated when the system resolves the addresses of neighboring systems connected with a broadcast network protocol such as Ethernet, the ARP MIB is not appropriate for use in Wide Area Networks, which typically use a non-broadcast protocol.

Short discovery for IPSec tunnels

The IP Manager reads the Routing Table of IPSec-enabled routers to discover the terminating endpoints (subinterfaces) of IPSec tunnels and IKE tunnels. It layers the subinterfaces over the underlying physical (WAN) interfaces and creates network connections between the local and remote subinterfaces to form the IPSec and IKE tunnels.

Monitoring and analysis

The IP Manager monitors the tunnels by monitoring their underlying WAN interfaces for Interface Down or DownOrFlapping events.

When the IP Manager detects an Interface Down or DownOrFlapping event on an underlying WAN interface, it places the host router on the short pending list for a short discovery (assuming that the autoReprobe_short parameter in the discovery.conf file is set to TRUE) for the following reason:

> IPSec and IKE tunnels are dynamic in the sense that they can be rerouted to other WAN interfaces when an interface fails. Therefore, when the IP Manager detects the Interface Down or DownOrFlapping event, it has no way of knowing whether the relationship between the tunnel and the underlying WAN interface, as known to the IP Manager, is still valid.

By rediscovering the host device, the IP Manager is able to recreate the tunnel layering over the WAN interface and provide accurate LayeredOver/Underlying relationships in the topology.

Note: The short pending list and the Pending Devices list are totally different lists.

Short discovery characteristics

Short discovery has the following characteristics:

- The short discovery pending list is consulted frequently (every 15 minutes by default).
- Short discovery is performed only on routers already in the topology.
- Short discovery changes only the attributes of already discovered router objects.
- Short discovery does not perform post-processing or codebook recomputation.

To change the short pending discovery interval, add the following configurable parameter to the discovery.conf file and modify its setting.

Parameter	Description	Default value
reprobePeriod_short	The time period before the next short discovery is triggered.	900 seconds (equates to 15 minutes)

Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* provides instructions on adding parameters to the discovery.conf file.

3

Name Resolution

This chapter describes the IP Manager name resolution process and provides procedures for configuring name resolution. It consists of the following sections:

- Discovery and the local name resolution service 3-2
- Hostname resolution and IP address checking 3-3
- How discovery names a system 3-4

Discovery and the local name resolution service

The discovery process relies on the name resolution service (a Domain Name System server, for example) provided by the local operating system to complete the following three tasks:

- For manual discovery, translate the hostnames of discovery candidate systems to corresponding IP addresses so that the systems can be discovered
- For manual discovery, translate the IP addresses of discovered systems to corresponding hostnames so that the discovered systems are given unique names within the topology
- For autodiscovery, translate the IP addresses of discovered systems to corresponding hostnames so that (1) the hostnames can be matched against the "System Name" field in the discovery filters (described in Chapter 7, “Creating Discovery Filters”) and (2) the discovered systems passing the discovery filters are given unique names within the topology

Until a system is discovered, it is known to the discovery process by its IP address only. When the system is discovered, the discovery process makes calls to the local name resolution service and runs a script to assign the system a distinct and meaningful name.

Hostname resolution and IP address checking

The discovery of a candidate system starts with a hostname, provided by the user, or an IP address, provided by the user or automatically discovered. If the system is identified by a hostname, the discovery process calls the local name resolution service to resolve the hostname to one or more IP addresses. If the hostname does not resolve, the discovery process places the system, identified by its hostname, on the Pending Devices list.

Note: The Pending Devices list is described in "Pending Devices list" on page 8-12.

If the candidate system is identified by a user-provided IP address, the discovery process checks whether the IP address conforms with the IP address format. If the IP address does not conform, the discovery process places the system, identified by its invalid IP address, on the Pending Devices list.

If the candidate system is identified by an automatically discovered IP address, the discovery process checks whether the IP address matches an IP address of any discovery filter. If the IP address does not match the IP address of any discovery filter, the discovery process discards the IP address.

How discovery names a system

When a system is discovered, the Name Resolution probe runs a script to determine a name for the system. The goal is to:

- Provide a unique name for the system.
- Use a consistent naming convention to ensure that the system is assigned the same name, if, for example, it is discovered using a different IP address.
- Update the name during a rediscovery if the name of the system has changed.

The name assigned to a system is determined by the setting of the NameFormat parameter in the name-resolver.conf file, which is located in the BASEDIR/smarts/conf/discovery directory. The assigned name becomes the value of the system's Name attribute.

Note: The values of a system's Name attribute and DisplayName attribute may be different, depending on the setting of the DisplayNameFormat parameter in the discovery.conf file. Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in the discovery.conf file and provides instructions on modifying the parameters.

The NameFormat parameter has two options:

- TM_USEAUTONAME (default)
- TM_USESEEDNAME

The options control how the Name Resolution probe assigns a name to a discovered system.

TM_USEAUTONAME option for automatic name resolution

When NameFormat is set to TM_USEAUTONAME, the Name Resolution probe makes calls to the local name resolution service and runs a script that resolves a name automatically by considering the following naming sources:

- Loopback IP address
- MIB-II sysName
- Public IP address
- Private IP address
- SNMP agent address

By default, the naming sources are considered in the order shown. You can change this order to create your own prioritized list of naming sources.

The procedure for configuring this option is described in "Configuring automatic name resolution" on page 3-6.

TM_USESEEDME option for seed-file name resolution

When NameFormat is set to TM_USESEEDNAME, the Name Resolution probe uses the following keyword and value defined for individual systems in the IP Manager's seed file to name the systems:

SEEDNAME=<user-specified name>

If this keyword is not defined for a system, the Name Resolution probe uses the original seed name—the name by which the system was introduced to the discovery process—to name the system. For an autodiscovered system, the seed name is the first non-private IP address on the system. The TM_USESEEDNAME option is useful when the topology information is provided by an external source.

The procedure for configuring this option is described in "Configuring seed-name name resolution" on page 3-9.

Note: Seed files are described in Chapter 6, "Preparing a Seed File."

Local exceptions to automatic and seed-file name resolution

For both automatic and seed-file name resolution, you can define local exceptions for individual systems in the seed file. Table 3-1 identifies the keyword=<value> exceptions.

Table 3-1 Overview of local name resolution exceptions for individual systems in a seed file

Automatic name resolution*	Seed-name name resolution**
Individual automatic name resolution exception using a prioritized list of naming sources: Not applicable	Individual automatic name resolution exception using a prioritized list of naming sources: • NAMEFORMAT=TM_USEAUTONAME The resolution process is performed in accordance to the naming source order (0 -> 4) specified by the AutoNameOrder entries in the name-resolver.conf file. The default order is: • AutoNameOrder 0 TM_USELOOPBACK • AutoNameOrder 1 TM_USESYSNAME • AutoNameOrder 2 TM_USENONPRIVATEIP • AutoNameOrder 3 TM_USEPRIVATEIP • AutoNameOrder 4 TM_USEAGENTADDRESS if you prefer a different naming source order, edit the name-resolver.conf file and change the AutoNameOrder numbers.
Individual automatic name resolution exceptions using an individual naming source: • NAMEFORMAT=TM_USELOOPBACK • NAMEFORMAT=TM_USESYSNAME • NAMEFORMAT=TM_USENONPRIVATEIP • NAMEFORMAT=TM_USEPRIVATEIP • NAMEFORMAT=TM_USEAGENTADDRESS	Individual automatic name resolution exceptions using an individual naming source: • NAMEFORMAT=TM_USELOOPBACK • NAMEFORMAT=TM_USESYSNAME • NAMEFORMAT=TM_USENONPRIVATEIP • NAMEFORMAT=TM_USEPRIVATEIP • NAMEFORMAT=TM_USEAGENTADDRESS
Individual seed-file name resolution exception specifying a specific name: • NAMEFORMAT=TM_USESEEDNAME SEEDNAME=<user-specified name>	Individual seed-file name resolution exception specifying a specific name: Not applicable

* NameFormat = "TM_USEAUTONAME" in name-resolver.conf file
** NameFormat = "TM_USESEEDNAME" in name-resolver.conf file

Note: For an ICMPONLY system, if automatic name resolution is in effect, use the NAMEFORMAT=TM_USESEEDNAME and SEEDNAME=<user-specified name> exception to name the system; if seed-file name resolution is in effect, use SEEDNAME=<user-specified name> to name the system.

The procedure for defining local exceptions to automatic name resolution is given in "Configuring exceptions to automatic name resolution" on page 3-8.

The procedure for defining local exceptions to seed-name name resolution is given in "Configuring exceptions to seed-name name resolution" on page 3-10.

Configuring automatic name resolution

Using the TM_USEAUTONAME option in the name-resolver.conf file, you can configure the Name Resolution probe to automatically resolve system names using the naming sources in the order that you specify. By default, the Name Resolution probe attempts to resolve system names using the sequence of naming sources shown in Figure 3-1.

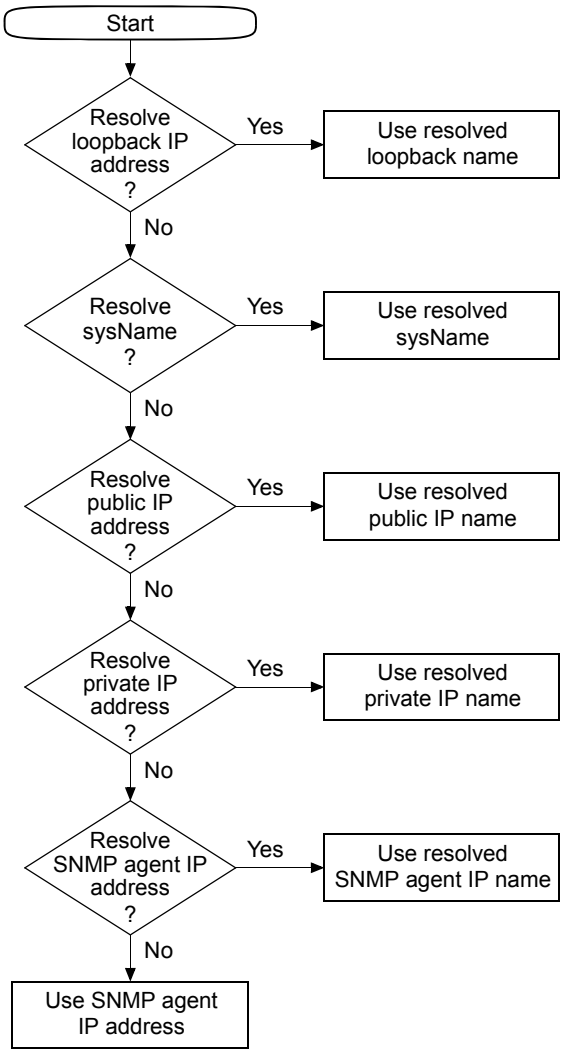


Figure 3-1 Default order of naming sources when automatically determining a system's name

To configure automatic name resolution:

1. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the *name-resolver.conf* file:

```
sm_edit conf/discovery/name-resolver.conf
```

2. Find the **NameFormat** section of the file:

```
#NameFormat = "TM_USESEEDNAME"
NameFormat = "TM_USEAUTONAME"
```

3. Ensure that the **NameFormat = "TM_USEAUTONAME"** option is the only one that is *not* commented out.

4. Find the **AutoNameOrder** section of the file:

```
AutoNameOrder 0 TM_USELOOPBACK
AutoNameOrder 1 TM_USESYSNAME
AutoNameOrder 2 TM_USENONPRIVATEIP
AutoNameOrder 3 TM_USEPRIVATEIP
AutoNameOrder 4 TM_USEAGENTADDRESS
```

5. Change the **AutoNameOrder** numbers (0 through 4) to the order that you prefer. The resolution process is performed in ascending order, starting with **AutoNameOrder 0**. Table 3-2 describes the **AutoNameOrder** naming sources.

Table 3-2 AutoNameOrder naming sources

Naming source	Description
TM_USELOOPBACK	Resolve loopback IP address: If a loopback interface IP address can be resolved to a name, that name will be used to name the system. By default, the default loopback interface type number is 24 (represents the "software loopback" interface type).
TM_USESYSNAME	Resolve MIB-II sysName: If the sysName can be resolved to at least one IP address and at least one of the resolved IP addresses resolves back to the sysName, the sysName will be used to name the system.
TM_USENONPRIVATEIP	Resolve non-private IP address: If at least one of the IP addresses is not a private IP address, the non-private IP address will be used to resolve the name. Once a name is resolved, the name will be used to name the system.
TM_USEPRIVATEIP	Resolve private IP address: If at least one of the IP addresses is a private IP address, the private IP address will be used to resolve the name. Once a name is resolved, the name will be used to name the system. By default, the private IP address pattern is 10.*|192.168.*|172.<16-31>.*, which is defined as follows: • Any IP address with the value of 10 in the first octet (10.*.*.*). • Any IP address with a value of 172 in the first octet and a value between 16 and 31 in the second octet (172.<16-31>.*.*). • Any IP address with a value of 192 in the first octet and a value of 168 in the second octet (192.168.*.*).
TM_USEAGENTADDRESS	Resolve SNMP agent address: If an SNMP agent address can be resolved to a name, that name will be used to name the system.

6. Save and close the *name-resolver.conf* file. The modified version of the file is saved to the *BASEDIR/smarts/local/conf/discovery* directory.

7. Issue the following command from *BASEDIR/smarts/bin* to make your changes take effect:

```
sm_tpmgr -s <IP Manager instance name> [-b <broker>]
   --load-conf=name-resolver.conf
```

For example:

```
sm_tpmgr -s INCHARGE-AM --load-conf=name-resolver.conf
```

Note: The option -b <broker> is used to specify the host and port (<host:port>) location of the Broker if the IP Manager is attached to a Broker other than "localhost:426."

8. If necessary, configure exceptions to automatic name resolution using a seed file, as described in "Configuring exceptions to automatic name resolution" on page 3-8.

During the next discovery or autodiscovery, name resolution will be active. If the resolution process is unsuccessful for all naming sources defined in the name-resolver.conf file, a final procedure uses a public IP address, a private IP address, or the SNMP agent address for the system name.

Configuring exceptions to automatic name resolution

You can define local exceptions to automatic name resolution by adding the keywords and values identified in Table 3-1 on page 3-5 to specific systems in the seed file. Chapter 6, "Preparing a Seed File," describes the format of a seed file.

To configure exceptions for automatic name resolution:

1. Configure automatic name resolution as described in "Configuring automatic name resolution" on page 3-6.
2. Create a seed file and, for each exception to the sources defined for automatic name resolution, add an entry for the system with either a specific naming source or a specific name:
   - To define a specific naming source for a system, add the following keyword and value to the seed file entry:

     ```
     NAMEFORMAT=<naming source>
     ```

     Where <**naming source**> is one of the naming sources in Table 3-2 on page 3-7. For example:

     ```
     192.168.12.2 NAMEFORMAT=TM_USELOOPBACK
     192.168.2.2 NAMEFORMAT=TM_USEPRIVATEIP
     ```

   - To define a specific name for a system, add the following keywords and values to the seed file entry:

     ```
     NAMEFORMAT=TM_USESEEDNAME SEEDNAME=<name>
     ```

     Where <**name**> is the specific name to use for the system. For example:

     ```
     ROUTER2 NAMEFORMAT=TM_USESEEDNAME SEEDNAME=Core-router-2
     192.168.8.29 NAMEFORMAT=TM_USESEEDNAME SEEDNAME=My-router
     ```

   Note that many other keywords can be used in the seed file in addition to the ones shown in the examples. Chapter 6, "Preparing a Seed File," identifies all of the keywords that can be used in a seed file.
3. Import the seed file for discovery. The locally specified naming sources and names will be used to name the systems. If the resolution process is unsuccessful for the specified naming source for a system, a final procedure will use a public IP address, a private IP address, or the SNMP agent address for the system name.

Configuring seed-name name resolution

When you select the TM_USESEEDNAME option in the name-resolver.conf file, the Name Resolution probe uses the following keyword and value defined for individual systems in the seed file to name the systems:

SEEDNAME=<user-specified name>

For any system in the seed file for which this keyword is not defined, the Name Resolution probe uses the seed name—the name by which the system was introduced to the discovery process—to name the system. For an autodiscovery candidate system, for which a seed name is not available, the Name Resolution probe uses the first non-private IP address to name the system.

To configure seed-name name resolution:

1. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the *name-resolver.conf* file:

   ```
   sm_edit conf/discovery/name-resolver.conf
   ```

2. Find the **NameFormat** section of the file:

   ```
   NameFormat = "TM_USESEEDNAME"
   #NameFormat = "TM_USEAUTONAME"
   ```

3. Ensure that the **NameFormat = "TM_USESEEDNAME"** option is the only one that is *not* commented out.

4. Save and close the *name-resolver.conf* file. The modified version of the file is saved to the *BASEDIR/smarts/local/conf/discovery* directory.

5. Issue the following command from *BASEDIR/smarts/bin* to make your changes take effect:

   ```
   sm_tpmgr -s <IP Manager instance name> [-b <broker>]
      --load-conf=name-resolver.conf
   ```

   For example:

   ```
   sm_tpmgr -s INCHARGE-AM --load-conf=name-resolver.conf
   ```

   During the next discovery, seed-name name resolution will be active.

6. Create a seed file and define a specific name for each system by adding the following to each entry:

   ```
   SEEDNAME=<name>
   ```

   Where <**name**> is the specific name to use for the system. For example:

   ```
   ROUTER2 SEEDNAME=Core-router-2
   192.168.8.29 SEEDNAME=My-router
   ```

   Note that many other keywords can be used in the seed file. Chapter 6, "Preparing a Seed File," identifies all of the keywords that can be used in a seed file.

7. If necessary, configure exceptions to seed-file name resolution, as described in "Configuring exceptions to seed-name name resolution" on page 3-10.

8. Import the seed file for discovery. The locally specified names for systems will be used to name the systems.

Configuring exceptions to seed-name name resolution

You can define local exceptions to seed-name name resolution by adding the keywords and values identified in Table 3-1 on page 3-5 to specific systems in the seed file. Chapter 6, "Preparing a Seed File," describes the format of a seed file.

To configure exceptions for seed-name name resolution:

1. Configure seed-name name resolution as described in "Configuring seed-name name resolution" on page 3-9.
2. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the *name-resolver.conf* file:

   ```
   sm_edit conf/discovery/name-resolver.conf
   ```

3. Find the **AutoNameOrder** section of the file:

   ```
   AutoNameOrder 0 TM_USELOOPBACK
   AutoNameOrder 1 TM_USESYSNAME
   AutoNameOrder 2 TM_USENONPRIVATEIP
   AutoNameOrder 3 TM_USEPRIVATEIP
   AutoNameOrder 4 TM_USEAGENTADDRESS
   ```

4. Change the **AutoNameOrder** numbers (0 through 4) to the order that you prefer. The resolution process is performed in ascending order, starting with **AutoNameOrder 0**. Table 3-2 on page 3-7 describes the **AutoNameOrder** naming sources.
5. Save and close the *name-resolver.conf* file. The modified version of the file is saved to the *BASEDIR/smarts/local/conf/discovery* directory.
6. Issue the following command from *BASEDIR/smarts/bin* to make your changes take effect:

   ```
   sm_tpmgr -s <IP Manager instance name> [-b <broker>]
      --load-conf=name-resolver.conf
   ```

   During the next discovery, the **AutoNameOrder** naming sources will be active and used when required.
7. Revise your seed file and, for each exception entry in the seed file, add a keyword and value as follows:

   ```
   NAMEFORMAT=<naming source>
   ```

   Where **<naming source>** is one of the naming sources identified in Step 3. For example:

   ```
   192.168.6.5 NAMEFORMAT=TM_USEAUTONAME
   192.168.12.2 NAMEFORMAT=TM_USELOOPBACK
   192.168.2.2 NAMEFORMAT=TM_USEPRIVATEIP
   ```

   Note that many other keywords can be used in the seed file in addition to the ones shown in the example. Chapter 6, "Preparing a Seed File," describes the format of a seed file.
8. Import the seed file for discovery. The locally specified naming sources will be used to name the systems. If the resolution process is unsuccessful for the specified naming source for a system, a final procedure will use a public IP address, a private IP address, or the SNMP agent address for the system name.

4

Preparing and Initiating Discovery

This chapter describes how to prepare for and initiate IP Manager discovery for manual discovery, autodiscovery, and topology import. It consists of the following sections:

- Preparing for manual discovery 4-2
- Preparing for autodiscovery 4-3
- Preparing for topology import 4-7
- Initiating discovery 4-8
- Initiating discovery for topology import 4-11
- Stopping autodiscovery 4-12

Preparing for manual discovery

Preparing the IP Manager for discovery using any method of discovery—manual, autodiscovery, or topology import—starts with seed systems, which serve as the starting point for building the IP Manager topology. Routers and switches are good seed system candidates because their MIBs contain many addresses and much information about additional systems in the network.

Using manual discovery alone, the IP Manager will only discover the seed systems. The seed systems are presented to the IP Manager as entries in a seed file, rather than being presented individually using the Add Agent command. A comprehensive seed file can only be created if the network topology information is complete and accessible.

In situations where the network topology information is incomplete, unavailable, or constantly changing, manual discovery is combined with autodiscovery. The seed file can be much smaller because the seed systems prepared for manual discovery also serve as a starting point for autodiscovery. Autodiscovery automatically discovers the managed network from the seed systems.

For manual discovery or manual discovery combined with autodiscovery, you create a seed file by editing the BASEDIR/smarts/conf/seedfile template in the IP Management Suite installation area. Chapter 6, “Preparing a Seed File,” includes the procedure for creating seed files.

Preparing for autodiscovery

Preparing for autodiscovery consists of the following tasks:

- "Create a seed file to seed the topology" (or use Add Agent command)
- "Create one or more discovery filters" (mandatory)
- "Specify autodiscovery safeguards" (optional)
- "Select autodiscovery data sources" (optional)
- "Enable autodiscovery" (mandatory)

The following sections provide procedures for completing these tasks.

In the discussions, you will see several references to the Pending Devices list and the Discovery Progress window, and about pending discovery and full discovery. "Pending Devices list" on page 8-12 provides information about the Pending Devices list that appears in the Discovery Progress window, and Chapter 10, "Scheduling and Manually Invoking Discovery," provides information about pending discovery and full discovery.

Create a seed file to seed the topology

Autodiscovery uses the IP addresses and topology gathered from seed systems to discover all of the systems in the managed network. The IP addresses or system names of the seed systems are specified in a seed file or in Add Agent commands.

Chapter 6, "Preparing a Seed File," includes the procedure for creating seed files.

Create one or more discovery filters

For autodiscovery and topology import, discovery filters determine which systems are discovered and added to the topology. You must create at least one discovery filter for autodiscovery.

Chapter 7, "Creating Discovery Filters," includes the procedure for creating discovery filters.

Specify autodiscovery safeguards

You can use one or both of the following safeguards to ensure that autodiscovery does not add systems (to the topology) that you do not intend to manage:

- "Using manual accept mode"
- "Setting a current system limit for the topology"

Using manual accept mode

You can configure a discovery filter for manual accept mode so that a candidate system matching the filter is added to the Pending Devices list in the THROTTLED state. By default, a candidate system that matches a filter is automatically discovered and added to the topology.

To enable manual accept mode for a filter:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)

2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Discovery Filters** tab.
4. Select the filter for which you want to enable the manual accept mode.
5. Click the **Ask before adding new systems** checkbox to enable manual accept mode for the filter.
6. Click **Apply**.

After autodiscovery is enabled and initiated, you can see what systems passed the filter and are waiting to be accepted by viewing the Pending Devices list. Systems that passed the filter and are waiting to be accepted are marked as THROTTLED. Accepting a THROTTLED system changes the state of the system to UNDISCOVERED; the system will be discovered during the next pending discovery interval.

Setting a current system limit for the topology

You can limit the number of systems that are added to the topology by setting a current system limit. By default, the current system limit is set to 50 systems.

To change the current system limit:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. In the **Current System Limit** field, enter the total number of systems that you want to allow into the topology.
5. Click **Apply**.

Note: The current system limit applies to all discovery methods: manual discovery, autodiscovery, and topology import.

When the current system limit is exceeded, the discovery process will complete the discovery of any system (and its components) in the midst of being discovered, including all virtual routers configured for a router or switch that implements virtual routers. When a router or switch implementing virtual routers is discovered, the chassis-based router/switch is counted as a system, and each virtual router is counted as a system. Any new discovery candidate systems will be added to the Pending Devices list in the UNDISCOVERED state, and the message "System limit exceeded" will appear in the Discovery Status section of the Discovery Progress window.

Select autodiscovery data sources

The IP Manager probes the topology and IP address table MIBs of the seed systems to find IP addresses of additional systems. Those IP addresses become candidate systems for the autodiscovery process.

The autodiscovery process can use three different data sources of information to discover candidate systems:

- "Topology MIBs"
- "IP address table MIB"
- "ARP table MIB"

Note: These data sources are described in "Autodiscovery data sources" on page 2-19.

By default, discovery uses the topology and IP address table MIBs of the discovered systems (initially, of the discovered seed systems) because the MIBs are very reliable sources for topology information.

To select which data source or sources are used for autodiscovery:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Select the data sources used by autodiscovery to find additional candidate systems. The data sources are listed underneath the **Enable Auto Discovery** checkbox.
5. Click **Apply**.

Topology MIBs

If you disable the topology MIB data source, autodiscovery may not be able to find all the systems in the managed network.

IP address table MIB

By default, the IP Manager pings the addresses in networks with an 8-bit host address (254 hosts), to prevent the indiscriminate pinging of large numbers of addresses. If necessary, you can change the default 8-bit host address setting to a larger value by editing the MaximumHostBits parameter in the discovery.conf file. Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in the discovery.conf file and provides instructions on modifying the parameters.

Note: It can take the IP Manager a long time, depending on your network configuration, to ping the entire list of addresses when the MaximumHostsBits is set to a large value, such as 16 bits (more than 65,000 addresses). With a value of 24 bits, for example, it could take the IP Manager up to several hours to ping all the available addresses.

ARP table MIB

By default, this data source is not enabled because autodiscovery does not typically gain many IP addresses using the ARP table MIB. In addition, autodiscovery does not read the ARP table MIB in hosts.

Enable autodiscovery

To enable autodiscovery:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Click the **Enable Auto Discovery** checkbox.
5. In the **Community String** field, provide any additional read community strings necessary to communicate with systems on your network. You may add up to four read community strings; the default is **public**.

   The community strings specified here are global default community strings that apply to autodiscovery, manual discovery, and topology import. "Adding and removing read community strings" on page 11-2 provides the procedure for adding community strings.
6. Click **Apply**.

Preparing for topology import

Topology import allows the IP Manager to discover a set of systems imported from a third-party source through the EMC Smarts IP adapter. The IP adapters are described in the *EMC Smarts IP Adapters User's Guide*.

Preparing for topology import consists of the following tasks:

- Modify discovery parameters in the discovery.conf and name-resolver.conf configuration files (mandatory)
- Modify the configuration file named server.conf (optional)
- Create a discovery filter for the IP Manager (mandatory)
- Specify a current system limit for the IP Manager (optional)

The *EMC Smarts IP Adapters User's Guide* provides procedures for completing these tasks.

You should create an open discovery filter that allows a broad range of systems to be added to the topology. Chapter 7, "Creating Discovery Filters," provides the information needed to create discovery filters. For an example, see "Filter for EMC Smarts IP adapter" on page 7-8.

Keep in mind that the open discovery filter created for topology import will adversely affect the discovery filters created for autodiscovery in the following regard: No matter where you place the topology import filter in the high-to-low priority stack of discovery filters, it will allow all autodiscovery candidate systems to pass through and be added to the discovery queue. Accordingly, you should consider disabling autodiscovery during the topology import process.

After the topology import process completes, you might want to enable autodiscovery and manually initiate a full discovery to autodiscover the neighboring systems of the imported systems added to the topology. Chapter 10, "Scheduling and Manually Invoking Discovery," includes the procedure for manually initiating a full discovery.

One final note: You can use the topology import method to import systems having SNMPv1 or v2c agents, but not v3 agents. You should list the systems having SNMPv3 agents in a seed file and use manual discovery to add them to the modeled topology. Chapter 6, "Preparing a Seed File," includes the procedure for adding SNMPv3 seed entries to a seed file.

Initiating discovery

The IP Manager provides two options for initiating a manual discovery. Both options are initiated through the Domain Manager Administration Console.

- Option 1: Import a seed file that lists one or more seed systems to be added to the topology. This option is best for adding a set of systems.
- Option 2: Add seed systems to the topology one at a time using the Add Agent command. This option is best for adding a small number of systems.

When autodiscovery is enabled, using either option initiates the autodiscovery process. When enabled and initiated, autodiscovery will occur whenever either a full discovery occurs or a system on the Pending Devices list is successfully discovered.

Using the import from seed file command to initiate discovery

To import a list of seed systems from a seed file:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. On the Domain Manager Administration Console, select **Topology > Import from seed file**, or click the **Import from seed** file toolbar button, to launch the **Import From Seed File** dialog box.
3. In the dialog box, specify the complete path and name of the seed file. By default, the path to the seed file is *BASEDIR/smarts/local/conf*. Note that the seed file must be stored on the host where the IP Manager runs.
4. Click **OK** to import the seed systems and to start the initial full discovery.

   A Discovery Progress window opens, displays progress messages, and might display a list of systems (pending systems) that could not be discovered. When you see "Last discovery completed" near the end of the progress report in the Discovery Status section, the probing of topology information is complete.

   Chapter 8, "Understanding Discovery Results," provides information about discovery errors and their solutions, and "Pending Devices list" on page 8-12 provides information about pending systems.
5. Click **Close** to close the Discovery Progress window.

Using the Add Agent command to initiate discovery

Note: Be careful not to add an overlapping IP address as a discovery candidate system. Overlapping addresses are discussed in "Discovery and IP tagging" on page 1-8.

Adding an SNMPv1/v2c seed system

To add an SNMPv1 or v2c seed system:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. On the Domain Manager Administration Console, select **Topology > Add Agent**, or click the **Add Agent** toolbar button, to display the **Add Agent** dialog box.

3. In the dialog box, specify the hostname or IP address of the v1/v2c seed system in the **Agent Name** field.

   You can also specify the port on which the SNMP agent of the seed system receives requests by typing a colon (:) and the port number after the hostname or IP address. If you do not specify a port, the default port specified in the *discovery.conf* file is used.

4. Optional: If the read community string of the v1/v2c seed system is not one of the strings in the **Read Community Strings** window (**Topology** tab) of the Domain Manager Administration Console, enter a read community string in the **Community String** field. "Adding and removing read community strings" on page 11-2 provides more information about read community strings.

5. Optional: Check the **Advanced Options** checkbox to display the **Add Agent Advanced Options** panel.

   In the **Advanced Options** panel, you can optionally specify the SNMP port, the SNMP version, and the access mode for the v1/v2c seed system. "Seed file format" on page 6-3 provides a description of these options and their values.

6. Click **OK** to add the v1/v2c seed system and to start the discovery process. The **Discovery Progress** window appears.

   Chapter 8, "Understanding Discovery Results," provides information about discovery errors and their solutions, and "Pending Devices list" on page 8-12 provides information about pending systems.

Adding an SNMPv3 seed system

To add an SNMPv3 seed system:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)

2. On the Domain Manager Administration Console, select **Topology > Add Agent**, or click the **Add Agent** toolbar button, to display the **Add Agent** dialog box.

3. In the dialog box, specify the hostname or IP address of the v3 seed system in the **Agent Name** field.

   You can also specify the port on which the SNMP agent of the seed system receives requests by typing a colon (:) and the port number after the hostname or IP address. If you do not specify a port, the default port specified in the *discovery.conf* file is used.

4. Check the **Advanced Options** checkbox to display the **Add Agent Advanced Options** panel.

   In the **Advanced Options** panel, you can optionally specify the SNMP port and the access mode for the v3 seed system. "Seed file format" on page 6-3 describes these options and their values.

5. In the **Advanced Options** panel, select SNMP version v3 to display the **Add Agent SNMPv3 Specifications** panel.

   In the **SNMPv3 Specifications** panel, you must specify the User name for the v3 seed system; you can optionally specify the SNMP engine ID, the authentication protocol, the privacy protocol, and the context name.

If you specify an authentication protocol (MD5, SHA, or NONE—NONE is the default), you must specify an authentication password; the password that you type is masked with asterisks (*) for security reasons. If you specify NONE for the authentication protocol or leave the field empty, no password is required.

If you specify a privacy protocol (DES, AES, or NONE—NONE is the default), you must specify a privacy password; the password that you type is masked with asterisks (*) for security reasons. If you specify NONE for the privacy protocol or leave the field empty, no password is required.

"Seed file format" on page 6-3 describes these options and their values.

6. Click **OK** to add the v3 seed system and to start the discovery process. The **Discovery Progress** window appears.

   Chapter 8, "Understanding Discovery Results," provides information about discovery errors and their solutions, and "Pending Devices list" on page 8-12 provides information about pending systems.

Initiating discovery for topology import

The IP Manager imports a set of seed systems from a third-party source through an EMC Smarts IP adapter when one of the following is initialized or restarted:

- IP Manager
- IP adapter
- Third-party source

When the IP adapter retrieves topology information from the third-party source, it produces a list of managed systems and their SNMP community strings for the IP Manager. For systems that match the discovery filter, the IP Manager adds them the managed topology or to the Pending Devices list.

The *EMC Smarts IP Adapters User's Guide* provides information about the operation of the EMC Smarts IP adapters.

Stopping autodiscovery

You can stop the autodiscovery process from the Discovery Progress window. (If the Discovery Progress window is not open, select Topology > Show Discovery Progress in the Domain Manager Administration Console to open the window.) In the Discovery Progress window, click Stop Autodiscovery to stop autodiscovery.

Clicking the Stop Autodiscovery button stops the finding of candidate systems for autodiscovery during Phase 3 of the discovery process. However, any systems already in the discovery queue continue to be processed, and the progress of this processing displays in the Discovery Progress window.

Stopping autodiscovery also disables future autodiscovery action until autodiscovery is re-enabled. To re-enable autodiscovery, see "Enable autodiscovery" on page 4-6.

5

Opening the Global Console

This chapter briefly describes the layout and features of the Global Console, including how to open the console. It consists of the following sections:

- Opening the Global Console 5-2
- Layout of the Domain Manager Administration Console 5-4

Opening the Global Console

The Global Console is a collection of many console views, two of which are the Domain Manager Administration Console and the Global Manager Administration Console. The consoles enable administrators to manage EMC Smarts Domain Managers and Global Managers.

User accounts and passwords

Attaching the Global Console to a Domain Manager (such as IP Availability Manager) or to a Global Manager requires an EMC Smarts user account. The default administration account is username admin and password changeme, and the default operator account is username oper and password oper.

For users logged on with an administration account, the Global Console includes a Configure menu, which enables administrators to open the Domain Manager Administration Console and the Global Manager Administration Console, to edit Map Icons, and so on.

For information about changing the password for the default administration account (recommended) and configuring access privileges, see the *EMC Smarts System Administration Guide*. For information about configuring permissions to perform specific console operations, see the *EMC Smarts Service Assurance Manager Configuration Guide*.

Procedure for opening the Domain Manager Administration Console

To open the Domain Manager Administration Console:

1. Start the Global Console.
   - On a Windows system, select **Start > Programs > InCharge 7> EMC Smarts Global Console**.
   - On a UNIX system, go to the *BASEDIR/smarts/bin* directory in the EMC Smarts Service Assurance Manager (Global Manager) installation area and enter **sm_gui**.

   The **Attach Manager** dialog box opens, an example of which is shown in Figure 5-1.

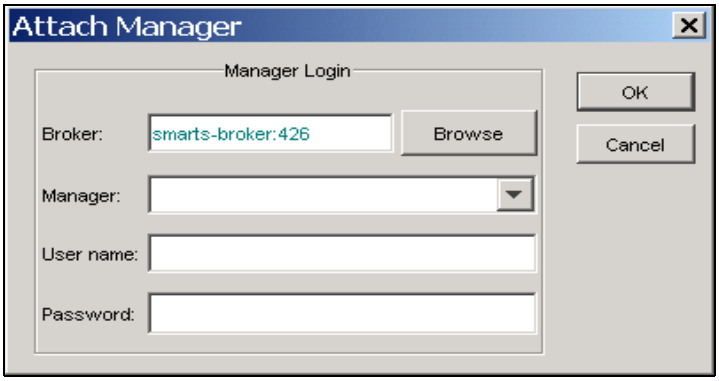


Figure 5-1 Attach Manager dialog box

2. In the dialog box,
   - Ensure that the EMC Smarts Broker for your deployment appears in the **Broker** text box.
   - Click the **Manager** drop-down list or the **Browse** button to display a list of active (running) Smarts Managers, and from that list select an IP Manager application (for example, INCHARGE-AM) in your deployment as the Smarts Manager to which you want to connect.
   - Enter your login username and password. (The default administration login is username *admin* and password *changeme*.)
   - Click **OK**.

   The **Topology Browser Console** opens.

   In the Console, select **Configure > Domain Manager Administration Console**. The **Domain Manager Administration Console** opens, an example of which is shown in Figure 5-2.

   In the example display, the **Domain Manager Administration Console** is attached to an IP Availability Manager application named INCHARGE-AM.

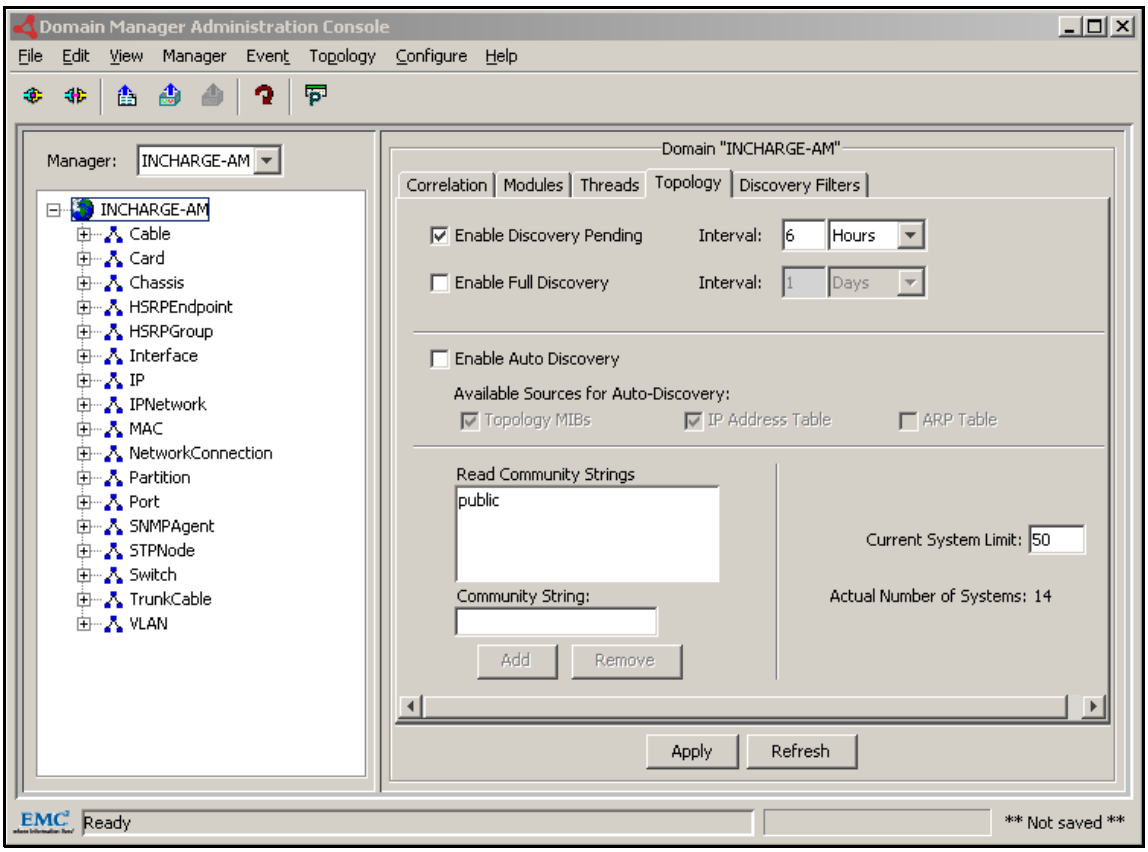


Figure 5-2 **Domain Manager Administration Console—example**

The Domain Manager Administration Console is the primary tool for configuring discovery, initiating discovery, and managing topology for a Domain Manager. The *EMC Smarts Service Assurance Manager Operator's Guide* provides detailed information about the Domain Manager Administration Console.

Layout of the Domain Manager Administration Console

The Domain Manager Administration Console (Figure 5-2) contains two panels. The left panel displays a topology tree that lists the attached Domain Managers and classes, or types, of managed objects.

When the Domain Manager is selected in the left panel, the right panel displays the following tabs:

- Correlation tab shows parameters related to correlation.
  - Correlation Interval determines how often the Domain Manager correlates events to find problems.
  - Codebook Radius determines the level of noise (delayed, lost, or spurious events) that the Domain Manager can tolerate and still guarantee correct correlation results.
  - Correlation Radius determines how exact of a match must occur between symptoms known to be caused by a problem and the symptomatic events received by the Domain Manager.
  - Number of Problems determines how many concurrent problems that the Domain Manager should consider possible when it correlates events.
  - Loss Symptom Probability is the likelihood that the Domain Manager will not receive notice of a symptom that actually occurred.
  - Spurious Symptom Probability is the likelihood that the Domain Manager will falsely receive notice that a symptom has occurred when it actually has not.
- Modules tab shows the libraries and programs that are loaded into the Domain Manager.
- Threads tab lists each running thread and its status.
- Topology tab shows the full discovery and pending discovery interval and provides controls for enabling regular full discoveries as well as autodiscovery. This tab is also where you specify default read community strings and set the system limit.
- Discovery Filters tab is where you specify filters for autodiscovery. At least one filter is required to enable autodiscovery.

When a class, such as Host, is selected in the left panel, the right panel displays the properties of the class under two tabs:

- Description tab provides a brief description of the class.
- Attributes tab lists attributes of the class. When a class is selected, the attributes do not contain values.

In addition, you can expand the classes listed in the left panel when there are instances of that class in the topology. Expanding a class lists the instances of that class. When an instance is expanded, the relationships that the instance participates in with other instances are revealed. Expanding the relationship shows the classes for which there are instances participating in the relationship.

When an instance (managed object) of a class is selected in the left panel, the right panel displays the properties of the instance under two tabs:

- Attributes tab lists the attributes and their values.
- Groups tab lists the Polling and Threshold groups to which the instance belongs plus the settings from those groups that apply to the instance.

Domain Manager Administration Console toolbar buttons

In the Domain Manager Administration Console, certain menu commands can be invoked through toolbar buttons. The toolbar buttons are summarized in Table 5-1.

Table 5-1 Domain Manager Administration Console toolbar buttons

Button	Description
	Attach to a Smarts Manager
	Detach from a Smarts Manager
	Import topology from a seed file
	Add a new SNMP agent to the topology
	Reconfigure: update instrumentation, recompute codebook, reapply polling/threshold values, and so on
	Open the Polling and Thresholds Console

6

Preparing a Seed File

This chapter describes how to prepare a seed file to provide discovery candidate systems to the IP Manager discovery process. It consists of the following sections:

- About seed files 6-2
- Seed file format 6-3
- Seed file examples for v1/v2c candidate systems 6-8
- Seed file examples for v3 candidate systems 6-9
- What systems to put in a seed file 6-10
- How to create a seed file 6-12
- How to encrypt passwords in the seed file 6-13

About seed files

Preparing a seed file is one of two ways to manually add systems to the IP Manager topology. The other way is using the Add Agent command, which is described in "Using the Add Agent command to initiate discovery" on page 4-8. When autodiscovery is enabled, importing a seed file or adding a candidate system using the Add Agent command initiates the autodiscovery process.

In a seed file, you specify top-level network systems such as switches, routers, hubs, bridges, and hosts that are to be discovered and monitored by the IP Manager. Then, when you import the seed file, the IP Manager discovers the internal structure of these systems.

For example, in the case of a router, you provide the hostname or IP address of one interface, and the IP Manager discovers the rest of the interfaces. In the case of a switch, you provide the hostname or IP address of the switch's SNMP agent, and the IP Manager determines the switch's configuration and how the switch connects to other systems listed in the topology.

Seed file template

The seed file must be stored on the host on which the IP Manager runs. By default, the IP Manager expects to find the seed file in the BASEDIR/smarts/local/conf directory. A template seed file named seedfile is provided in BASEDIR/smarts/conf.

Seed file dual use

The seed entries in the seed file are read by the IP Manager's topology manager and trap adapter at startup. The topology manager uses the entries and their credentials to contact the seed systems during discovery, and the trap adapter uses the entries and their credentials to verify and decrypt SNMPv3 trap messages received from SNMPv3-capable seed systems once they are discovered.

Seed file format

A seed file consists of one or more seed entries, each of which represents a discovery candidate system. Each seed entry begins with the hostname or IP address for the candidate system followed by options for the system. The allowed options are specified using the keywords described in Table 6-1.

Table 6-1 Keywords For seed file

Keyword	Value	Definition
Global SNMP Options		
ACCESSMODE=	ICMPONLY, SNMPONLY, or ICMPSNMP Default: ICMPSNMP	Protocols used to access the candidate system: • For ICMPSNMP, both ICMP and SNMP are used to discover and monitor the system. • For ICMPONLY, the system is not discovered or monitored. Instead, the system is added to the topology as a Host, which limits the IP Manager to only availability analysis using ICMP. • For SNMPONLY, the system is discovered using SNMP, but ICMP is not used to determine whether the system is reachable or to monitor the system's availability.
ADDRESSFORMAT=	AUTOADDRESS or LOOPBACKADDRESS Default: value in discovery.conf	Address used to access the candidate system for discovery and polling. If the value is omitted, the value specified in the AccessAddressFormat parameter (AUTOADDRESS by default) in the discovery.conf file is used. The *EMC Smarts IP Management Suite Configuration Guide* provides information about the AccessAddressFormat parameter.
SNMPPORT=	0 to 65535 Default: port in discovery.conf	SNMP port used to access the SNMP agent of the candidate system. If the value is omitted or 0, the port specified in the defaultSNMPPort parameter (161 by default) in the discovery.conf file is used. The *EMC Smarts IP Management Suite Configuration Guide* provides information about the defaultSNMPPort parameter,.
SNMPVERSION=	V1, V2C, V3, or AUTODETECT Default: AUTODETECT	SNMP protocol version supported by the SNMP agent of the candidate system: • For AUTODETECT, send v2c request; if no response, send v1 request. • For V1, send v1 request only. • For V2C, send v2c request only. • For V3, send v3 request only. The SNMP protocol version is also the expected version of any trap message received from the candidate system after the system is discovered. For SNMPVersion=AUTODETECT, the expected trap message version is v2c or v1, depending on the version of the first SNMP poll to successfully communicate with the SNMP agent of the candidate system.

* Specified in the Topology tab of the Domain Manager Administration Console.
** MD5 and SHA authentication are defined in RFC 3414.
*** Explained in "How to encrypt passwords in the seed file" on page 6-13.
**** DES privacy is defined in RFC 3414, and AES (AES-128) privacy is defined in RFC 3826.

Table 6-1 Keywords For seed file (continued)

Keyword	Value	Definition
NAMEFORMAT=	TM_USEAUTONAME, TM_USELOOPBACK, TM_USESYSNAME, TM_USENONPRIVATEIP, TM_USEPRIVATEIP, TM_USEAGENTADDRESS, or TM_USESEEDNAME Default: null string (empty)	Naming source used to resolve the name of the candidate system. "How discovery names a system" on page 3-4 provides information on the NAMEFORMAT and SEEDNAME keywords.
SEEDNAME=	String of unspecified length Default: null string (empty)	User-specified name for the candidate system.
SNMPv1 and v2c Options		
COMMUNITY=	String of unspecified length Default: read string or strings in Read Community Strings window *	Read community string included in a request sent to the SNMPv1/v2c agent of the candidate system. This keyword is required if the first character of the string is #.
SNMPv3 Options		
USER=	String consisting of 1 to 32 characters Default: null string (empty)	Name of the user included in a request sent to the SNMPv3 agent of the candidate system or received in an SNMPv3 trap message. User name is required.
ENGINEID=	An even-length string between 10 and 64 hex characters; e.g., 800002b804616263 Default: null string (empty)	Identifier, within an administrative domain, that uniquely identifies the SNMP engine of the SNMPv3 agent. Engine ID is required for communication to the SNMPv3 agent: • When specified, the IP Manager uses the specified value when attempting to communicate with the SNMP agent. • If empty, the IP Manager learns the engine ID through the engine ID discovery mechanism described in RFC 3414 and then uses the learned value when attempting to communicate with the SNMP agent. Engine ID is also required to authenticate and decrypt an SNMPv3 trap message.
AUTHPROTOCOL=	MD5, for Message Digest 5, SHA, for Secure Hash Algorithm, or NONE Default: NONE **	For authentication: • Protocol used for authentication when sending a request to the SNMPv3 agent. • Authentication protocol is recommended but not required. For authenticating: • Protocol used for authenticating an SNMPv3 trap message. • Authentication protocol is required to authenticate an SNMPv3 trap message.
AUTHPASS=	String consisting of 1 to 64 characters; using at least eight characters is recommended Default: null string (empty)	Password used as the basis for the creation of the localized authentication key used with the selected authentication protocol and the SNMPv3 agent. The password is required if the authentication protocol is MD5 or SHA. Otherwise, the password is ignored. The password may be encrypted (recommended) or appear as plain text in the seed file. ***

* Specified in the Topology tab of the Domain Manager Administration Console.
** MD5 and SHA authentication are defined in RFC 3414.
*** Explained in "How to encrypt passwords in the seed file" on page 6-13.
**** DES privacy is defined in RFC 3414, and AES (AES-128) privacy is defined in RFC 3826.

Table 6-1 Keywords For seed file (continued)

Keyword	Value	Definition
PRIVPROTOCOL=	DES, for Data Encryption Standard, AES, for Advanced Encryption Standard, or NONE Default: NONE ****	For encryption: • Protocol used for encryption when sending a request to the SNMPv3 agent. • Privacy protocol is recommended but not required.
		For decrypting: • Protocol used for decrypting an SNMPv3 trap message. • Privacy protocol is required to decrypt an SNMPv3 trap message.
PRIVPASS=	String consisting of 1 to 64 characters; using at least eight characters is recommended Default: null string (empty)	Password used as the basis for the creation of the localized encryption key used with the selected privacy protocol and the SNMPv3 agent. The password is required if the privacy protocol is DES or AES. Otherwise, the password is ignored. The password may be encrypted (recommended) or appear as plain text in the seed file. ***
CONTEXT=	String of no specific length Default: null string (empty)	Name that together with the User name determines the access permissions of a request sent to the SNMPv3 agent. Context name is optional. The IP Manager must have permissions to access the standard and proprietary MIBs identified in Appendix A of the *EMC Smarts IP Availability Manager User's Guide*.

* Specified in the Topology tab of the Domain Manager Administration Console.
** MD5 and SHA authentication are defined in RFC 3414.
*** Explained in "How to encrypt passwords in the seed file" on page 6-13.
**** DES privacy is defined in RFC 3414, and AES (AES-128) privacy is defined in RFC 3826.

Notation and syntax

You use <keyword>=<value> to specify options for a seed entry. (Blank spaces before and after the equal sign (=) are not permitted.) The <keyword>=<value> options for a seed entry may be arranged in any order, and they may span several lines for an entry. Here's an example of a seed entry beginning with an IP address:

```
20.30.4.1 SNMPPORT=200 COMMUNITY=p100 ACCESSMODE=ICMPSNMP
   SNMPVERSION=V2C
```

The first character of a comment line is the pound sign (#). For example:

```
# The seed file is used to import top-level network systems.
```

To make your seed file more readable, you can include blank lines and comments.

Specifying a port

For a seed entry, you can specify the port number of the SNMP agent running on the candidate system in one of two ways:

- By typing a colon (:) and the port number after the hostname or IP address; for example:

  ```
  CoreROUTER-2:200
  ```

- By using the SNMPPORT keyword described in Table 6-1; for example:

  ```
  CoreROUTER-2 SNMPPORT=200
  ```

Specifying a community string

For a candidate system having an SNMPv1 or v2c agent, the second field of a seed entry is the optional read community string. If you do not provide a community string for a v1 or v2c candidate system, the IP Manager uses the default read community string or strings specified in the Read Community Strings window (Topology tab) of the Domain Manager Administration Console—see "Adding and removing read community strings" on page 11-2 for details. If the community string starts with the pound sign (#), you must use the COMMUNITY keyword (Table 6-1) to specify the read community string.

Mandatory keyword options for SNMPv1 or v2c

For a candidate system having an SNMPv1 or v2c agent, no keyword options are mandatory because all global and SNMPv1/v2c keywords have default values that are applicable to an SNMPv1 or v2c agent. So, specifying just a hostname or an IP address for an SNMPv1/v2c seed entry will not cause the IP Manager to throw an exception when importing the seed file.

Mandatory keyword options for SNMPv3

For a candidate system having an SNMPv3 agent, two keyword options are mandatory:

- SNMPVERSION
- USER

If you do not specify both keyword options for an SNMPv3 seed entry, the IP Manager will throw an exception for the seed entry when reading the seed file.

To authenticate or decrypt an SNMPv3 trap, you must include an entry in the seed file that matches the source of the trap. When the IP Manager receives an SNMPv3 trap, it reads the SNMPv3 agent's User name and Engine ID, searches the seed file for an SNMPv3 seed entry match, and uses the credentials associated with that seed entry to authenticate or decrypt the trap message.

Table 6-2 identifies the required keyword options for various SNMPv3 seed entry scenarios.

Table 6-2 Required keyword options for various SNMPv3 seed entry scenarios

Seed entry scenario	Keyword options required
Seed entry 1: Send an authenticated message to an SNMPv3 agent:	• SNMPVERSION=V3 • USER=<user name> • AUTHPROTOCOL=MD5 or SHA • AUTHPASS=<auth password>
Seed entry 2: Send an encrypted message to an SNMPv3 agent:	• SNMPVERSION=V3 • USER=<user name> • PRIVPROTOCOL=DES or AES • PRIVPASS=<priv password>

Table 6-2 Required keyword options for various SNMPv3 seed entry scenarios (continued)

Seed entry scenario	Keyword options required
Seed entry 3: Send an authenticated and encrypted message to an SNMPv3 agent:	• SNMPVERSION=V3 • USER=<user name> • AUTHPROTOCOL=MD5 or SHA • AUTHPASS=<auth password> • PRIVPROTOCOL=DES or AES • PRIVPASS=<priv password>
Seed entry 4: Authenticate an SNMPv3 trap:	• SNMPVERSION=V3 • USER=<user name> • ENGINEID=<engine id> • AUTHPROTOCOL=MD5 or SHA • AUTHPASS=<auth password>
Seed entry 5: Decrypt an SNMPv3 trap:	• SNMPVERSION=V3 • USER=<user name> • ENGINEID=<engine id> • PRIVPROTOCOL=DES or AES • PRIVPASS=<priv password>
Seed entry 6: Authenticate and decrypt an SNMPv3 trap:	• SNMPVERSION=V3 • USER=<user name> • ENGINEID=<engine id> • AUTHPROTOCOL=MD5 or SHA • AUTHPASS=<auth password> • PRIVPROTOCOL=DES or AES • PRIVPASS=<priv password>

Seed file examples for v1/v2c candidate systems

The following examples of seed file entries illustrate the use of the keywords (listed in Table 6-1 on page 6-3) for an SNMPv1 or v2c candidate system.

The first seed entry spans multiple lines and specifies the SNMP version and the read community string.

```
Core-ROUTER-1 SNMPVERSION=V1
      COMMUNITY=xxxxyyyy#%@*&^)(:"=|\][}{
```

The second seed entry specifies the SNMP port (2161) that the IP Manager should use to access the system.

```
ROUTER2:2161 SNMPVERSION=V2C COMMUNITY=public1
```

The third seed entry requires the keyword COMMUNITY because the first character of the read community string is a pound sign (#).

```
router-2.example.com COMMUNITY=#abcdefg SNMPPORT=2222
```

The fourth seed entry can be specified without using keywords. The only two fields are the system identifier and the read community string. The COMMUNITY keyword is not required even though the first character of the community string is the equal sign (=).

```
192.168.2.2  =abcdefg
```

Seed file examples for v3 candidate systems

The following examples of seed file entries illustrate the use of the keywords, which are listed in Table 6-1 on page 6-3, for an SNMPv3 candidate system.

The first seed entry spans multiple lines and specifies the SNMP version, the username, and no authentication.

```
Core-SWITCH-1 SNMPVERSION=V3 USER=inch_n
      ENGINEID=800002b804616263
```

The second seed entry specifies the SNMP port (2161) that the IP Manager should use to access the system.

```
SWITCH2:2161 SNMPVERSION=V3 USER=inch_n
```

The third seed entry specifies that the default authentication protocol MD5 should be used to access the system. The prefix <E-1.0> indicates that the authentication password is encrypted.

```
switch-2.example.com SNMPVERSION=V3 USER=smarts SNMPPORT=2222
      ENGINEID=6acff2d4098338 AUTHPROTOCOL=MD5
      AUTHPASS=<E-1.0>869D64B72C38FE49E881BC38BCCE6A0B
      1DE597A5871636C4B0F139FA7EB1100C
```

The encrypted password, which wraps around in the display but is actually one line, was created using the procedure in "How to encrypt passwords in the seed file" on page 6-13.

The fourth seed entry specifies that the authentication protocol SHA should be used to access the system and that access control is in effect. The authentication password is not encrypted.

```
192.154.1.1 SNMPVERSION=V3 USER=smarts_2 AUTHPROTOCOL=SHA
      AUTHPASS=sally121#%
      CONTEXT=incharge_west
```

The fifth seed entry specifies that the authentication protocol SHA should be used to access the system and that the privacy protocol AES should be used to encrypt the SNMP messages sent to the system. The prefix <E-1.0> indicates that the authentication password and the privacy password are both encrypted.

```
128.221.19.8 SNMPVERSION=V3 USER=shaAesUser AUTHPROTOCOL=SHA
      AUTHPASS=<E-1.0>123ABC456#$%123abc456#$%
      PRIVPROTOCOL=AES
      PRIVPASS=<E-1.0>456123abc456#$%456123ABC456#$%
      ENGINEID=0000000902000003E333C440
```

What systems to put in a seed file

The seed file should list the top-level network systems such as routers, switches, hubs, bridges, and hosts that are to be managed by the IP Manager. Routers and switches are better choices than hosts for seed systems because their MIBs contain the IP addresses of many additional network systems.

Note: Be careful not to include overlapping IP addresses in the seed file. Overlapping addresses are discussed in "Discovery and IP tagging" on page 1-8.

Seed file sources

To create the seed entries in a seed file, you can usually find the information that you need from existing computer-accessible sources.

- If you are currently using CiscoWorks with Cisco Resource Manager, you can dump the hardware inventory to a text file. That text file can be converted into a seed file. Similar operations might be possible with other network management platforms.
- If you are currently using one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide*, you can obtain the network information from the ovsnmp.conf file. This configuration file is maintained by the xnmsnmpconf configuration utility.
- If you store backup router or switch configuration files at a central location, you can construct a seed file from them.
- Domain name resolution files, such as hosts.txt or zone files, are often a useful source. Note that, at sites where multiple administrative authorities choose different read community strings (applicable to SNMPv1 or v2c agents), the hosts listed in a given zone file often share an administrative authority and hence a read community string. Terminal Access Controller Access Control System (TACACS) configuration files might be useful for similar reasons.
- Large sites usually maintain hardware inventories in a database or at least in some computer-readable form. These inventories often contain the network information that you need to create the seed file.
- Some sites run network sniffer tools to keep track of active IP addresses. The lists of addresses that they generate can form the basis for a seed file.

Autodiscovery seed systems

If you plan on using autodiscovery, consider including a Cisco Discovery Protocol (CDP) enabled, Extreme Discovery Protocol (EDP) enabled, or Foundry Discovery Protocol (FDP) enabled system in the seed file when possible to improve autodiscovery coverage. The discovery process is able to probe CDP, EDP, FDP, and other tables, to take advantage of the abundance of topology information that they hold.

SNMPv3 systems

Autodiscovery cannot be used to discover systems that use SNMPv3 agents, and topology import (using one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide*) cannot be used to import SNMPv3 systems. To discover SNMPv3 systems, add them to the seed file.

Multiple seed entries for multiple IP-address systems

After you have identified which systems to include in the seed file, you might want to include multiple seed entries for systems having multiple IP addresses (routers, for example). The discovery process will realize that the multiple IP addresses actually belong to a single system. Accordingly, the IP Manager will be able to reach the system using an alternate address if the primary address is inaccessible (due to a failure) during the initial probe of the system. After a system is discovered and all its IP addresses are known, the IP Manager will automatically try up to ten different IP addresses to route around failures and reach the system.

How to create a seed file

You typically use the template file named seedfile, located in the BASEDIR/smarts/conf directory, as the basis for creating a seed file.

To create a seed file using the seedfile template:

1. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the *seedfile*:

   ```
   sm_edit conf/seedfile
   ```

2. Add the seed entries for the systems to be discovered.
3. Save and close the file. The modified version of the file is saved to the *BASEDIR/smarts/local/conf* directory.

How to encrypt passwords in the seed file

When viewing the seed file with the sm_edit utility, add the following line (including #) as the first line in the seed file to direct sm_edit to encrypt the authentication and privacy passwords in the file:

```
#<encrypted seed>:1.0:AUTHPASS,PRIVPASS
```

When you close the seed file, sm_edit will automatically encrypt the authentication and privacy passwords in the file.

To determine whether a password in a seed file has been encrypted, look for the prefix <E-1.0> in the password. If <E-1.0> is present, the password is encrypted; if <E-1.0> is not present, the password is plain text. Although seed files containing plain-text passwords are supported, they are not recommended.

The sm_edit utility does not encrypt a password that is already encrypted, determined by the presence of the <E-1.0> prefix. Once a password is encrypted, you cannot use sm_edit to decrypt the password. If you wish to change an encrypted password, you must delete the password (including the <E-1.0> prefix), enter the new password, and save the file.

During the installation of the IP Manager, the installation program uses the case-sensitive secret phrase *Not a secret* to create a secret key, to be used by sm_edit and other EMC Smarts utilities to encrypt passwords. The secret phrase and secret key, themselves, are encrypted and stored in the BASEDIR/local/conf/imk.dat file. After the installation, an EMC Smarts administrator should use the sm_rebond utility to change the secret phrase and secret key.

The *EMC Smarts System Administration Guide* provides information about sm_rebond and EMC Smarts encryption. The sm_edit utility is discussed in Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide*.

7

Creating Discovery Filters

This chapter describes how to create discovery filters to control what discovery candidate systems are added to the IP Manager discovery process. It consists of the following sections:

- About discovery filters 7-2
- Discovery filter fields 7-3
- Extended filter information 7-4
- Examples of discovery filters 7-7
- How to create a discovery filter 7-10

About discovery filters

The purpose of discovery filters is to control or limit what systems are added to the discovery process during autodiscovery. You define discovery filters to ensure that certain network-accessible systems not intended for management by the IP Manager are not discovered or placed on the Pending Devices list.

For example, an Internet service provider (ISP) might want to discover a router and its interfaces but not the systems connected to the router on the ISP's client side. Configuring the appropriate discovery filters will enforce this discovery restriction.

Because discovery filters are inclusive-type filters, meaning that an autodiscovery candidate system must match an IP address of a discovery filter before it is eligible for discovery, you must specify at least one discovery filter to use autodiscovery.

Appropriate discovery filtering ensures that autodiscovery is efficient. In addition, you can limit and control autodiscovery by:

- Specifying exclusion filters using the ipExcludeList parameter in the BASEDIR/smarts/conf/discovery/discovery.conf file.
- Specifying manual accept mode or setting a system limit as described in "Specify autodiscovery safeguards" on page 4-3.

Note: Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in the discovery.conf file and provides instructions on modifying the parameters.

Importing topology through one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide* also requires a discovery filter. In this case, however, because you are not trying to control or limit the discovery, you create a single, open discovery filter to accept all possible system names and addresses provided by the third-party source.

Note: Autodiscovery cannot be used to discover systems that use SNMPv3 agents, and topology import cannot be used to import SNMPv3 systems. To discover SNMPv3 systems, add them to the seed file.

Discovery filter fields

Because the IP Manager discovery is based on candidate systems represented as IP addresses, discovery filters are based on IP addresses. All that is known about a candidate system at the beginning of the autodiscovery process is its IP address.

You specify filters using the five discovery fields identified in Table 7-1.

Table 7-1 Fields of a discovery filter

Discovery filter fields	Discovery filter example
IP Address Range	* (matches any IP address)
System Name†	* (matches any system name)
System Description†	* (matches and system description)
SystemOID	* (matches any system OID)
System Types	Switch
† The System Name and System Description fields are case-sensitive.	

The discovery filter example shown in the table will match any candidate system that is a Switch. The asterisk (*) in the IP address range, system name, system description, and system OID fields is a wildcard character that matches any arbitrary string of characters. The string can be empty.

Discovery filters are inclusive, meaning that if you do not create any filters, no candidate systems are accepted. A system is accepted only if it matches a filter.

Consider the following points before creating discovery filters.

- When creating a filter, you must specify a value for each of its five fields. A candidate system must pass each field to match the filter.
- You can create multiple filters and arrange them in a high-to-low priority. A candidate system is compared against the highest priority filter first.
- A candidate system is subjected to two levels of filter matching:
  - A matching of IP address in the first field of a filter, to determine whether the system is eligible for discovery
  - A matching of other criteria in the remaining four fields of a filter after the candidate system passes ICMP/SNMP polling, to determine whether the system should be added to the topology.

  The second level of filter matching is described in "Extended filter information" on page 7-4.
- When a candidate system matches all five fields of a filter and passes any additional autodiscovery safeguards, it bypasses any lower priority discovery filters and becomes an object in the modeled topology.
- To exclude a specific candidate system, specify the system in the ipExcludeList in the discovery.conf file.
- Seed systems, as listed in a seed file or entered through the Add Agent command, are not subject to the discovery filters.

For systems that match a discovery filter and are added to the topology, they too are probed for the IP addresses of their neighboring systems. The autodiscovery cycle continues until no more new IP addresses match the discovery filters.

Extended filter information

In addition to the five fields of a discovery filter, you can specify additional information for a filter to control the polling of any system that matches an IP address of the filter. The additional fields, referred to as extended options, are identified in Table 7-2.

Table 7-2 Applying extended options to a discovery filter

Discovery filter fields
Core Fields
IP Address Range
System Name
System Description
SystemOID
System Types
Extended Option Fields
SNMP Port
SNMP Version*
Access Mode
Community String
* SNMPv3 is not currently supported for autodiscovery.

When a candidate system's IP address matches an IP address of an extended discovery filter, the IP Manager uses the IP address and the following user-specified extended options of the filter to perform the initial ICMP/SNMP polling of the candidate system (the polling is shown in the figure "Phase 1: Initial polling of a candidate system" on page 2-6):

- SNMP port number (161 by default)
- SNMP version (V1, V2, AUTODETECT; AUTODETECT by default)
- Access mode (ICMPSNMP or ICMPONLY; ICMPSNMP by default)
- Read community string (public by default)

For a successful SNMP polling, the IP Manager reads the SNMP response to learn the system name (sysName), system description (sysDescr), system OID (sysObjectID), and other system-related information of the candidate system.

The IP Manager then performs the following tasks to determine whether the candidate system matches the filter (the filter matching is shown in the figure "Phase 2: Determining the destination of a candidate system" on page 2-7):

- Compares the retrieved system description against the "System Description" field in the filter.
- Compares the retrieved system OID against the "SystemOID" field in the filter.

- Maps the retrieved system OID to an OID in the oid2type configuration files (described in Chapter 2, "Discovery Process") to identify the system's type, and then compares that type against the "System Types" field in the filter.
- Passes the candidate system's IP address to the local name resolution service (described in Chapter 3, "Name Resolution") to resolve the address to a name, and then compares that name against the "System Name" field in the filter.

If the candidate system matches the filter, it enters Phase 3 of the discovery process. If the candidate system does not match the filter, the system is compared against the core fields of the next lower priority filter. Whether the system matches this filter or another filter in the stack of discovery filters, the system will keep the extended options assigned to it during the initial ICMP/SNMP polling. If the system does not match any filter, it is discarded.

The two filters in Table 7-3 help clarify this behavior.

Table 7-3 Two discovery filters having extended options

Discovery filter fields	Filter 1 values	Filter 2 values
IP Address Range	*	*
System Name	*	*
System Description	*	*
SystemOID	*	*
System Types	Switch	Router
SNMP Port	165	0
SNMP Version†	AUTODETECT	V2C
Access Mode	ICMPSNMP	ICMPSNMP
Community String	public1	public2
† SNMPv3 is not currently supported for autodiscovery.		

The candidate system to be compared with Filters 1 and 2 is a router. Although the router matches Filter 2 but not Filter 1, the router keeps the Filter 1 extended options included in the initial SNMP poll of the system: the router's SNMP port is 165, its SNMP version is AUTODETECT, its access mode is ICMPSNMP, and its read community string is public1.

Note: For a candidate system that matches the IP address of a non-extended filter, the IP Manager uses the default values for the extended options when contacting the system.

Other important information relevant to extended options:

- SNMP Port enables you to specify an alternate port on which systems receive SNMP requests. The default value is 0, which indicates that the IP Manager uses the port number value (initially 161) of the defaultSNMPPort parameter in the discovery.conf file.
- SNMP Version enables you to control what version of SNMP that the IP Manager uses to communicate with the system. Valid values for autodiscovery are V1, V2C, and AUTODETECT. The default is AUTODETECT, meaning that discovery tries SNMPv2c first and then SNMPv1 if necessary. If you specify an explicit SNMP version, V2C or V1, the discovery process tries only the SNMP version that

you specified. If the explicit version is different from the actual version supported by the system, a DiscoveryError with a message of No response from SNMP Agent is notified.

- Access Mode determines what protocols are used to both discover and monitor the system and its components. Valid values for autodiscovery are ICMPSNMP and ICMPONLY. The default is ICMPSNMP, meaning that both ICMP and SNMP protocols are used to discover and manage the system. When ICMPONLY is selected, discovery does not send SNMP polls to the system and cannot, therefore, discover the system and its components; instead, the system is added to the topology as a Host and is monitored only for connectivity. Regardless of whether a system is added to the topology using ICMPSNMP or ICMPONLY, an associated SNMPAgent for the system is also added to the topology.

  Note: When an ICMPONLY host has multiple IP addresses that resolve to the same name, all IP addresses are retained. You must manually delete any IP addresses that no longer belong to the ICMPONLY host.

  Note: An additional Access Mode option, SNMPONLY, is available to a system imported from a seed file or added to the topology using the Add Agent command.

- Community String enables you to specify the one or more read community strings that should be used to contact systems that match this filter. If no community strings are specified, autodiscovery uses the default read community string. "Adding and removing read community strings" on page 11-2 provides information about default community strings.

Examples of discovery filters

A discovery filter can be broad, to discover a broad range of systems, or restrictive, to limit the number or type of discovered systems.

For a filter's IP address range, system name, system description, and system OID fields, you can use wildcards to create a matching pattern. Appendix C, "Wildcard Patterns," discusses wildcard syntax.

The following examples illustrate different types of discovery filters.

Filtering a range of IP addresses

To exclude a particular range of systems, for example 22 in the second octet of an IP address range, use a filter that includes every other possibility as shown in Table 7-4.

Table 7-4 Filtering by IP address

Discovery filter fields	Value
IP Address Range	172.<16-21>.* | 172.<23-31>.55.*
System Name	*
System Description	*
SystemOID	*
System Types	Select all types

Note: In general, you should use the ipExcludeList configuration parameter—not a discovery filter—to exclude specific systems. The ipExcludeList parameter is in the discovery.conf file.

Filtering by system type, name, and description

To narrow the range of a filter, specify criteria for multiple fields, as shown in Table 7-5. Only those systems that are routers, within a specified IP address range, whose system name begins with *ny*, and whose system description begins with *Cisco* match this filter.

Table 7-5 Filtering by system type, name, and description

Discovery filter fields	Value
IP Address Range	172.21.55.*
System Name	ny*
System Description	Cisco*
SystemOID	*
System Types	Router

Filtering with manual accept mode

To manually accept each candidate system, enable manual accept mode—described in "Using manual accept mode" on page 4-3. In the example shown in Table 7-6, routers and switches that match Filter 1 are automatically added to the topology. Hosts, however, that match Filter 2 are automatically added to the Pending Devices list.

Table 7-6 Filtering with accept mode

Discovery filter fields	Filter 1 values	Filter 2 values
IP Address Range	172.21.55.*	172.21.55.*
System Name	*	*
System Description	*	*
SystemOID	*	*
System Types	Routers Switches	Hosts
Ask before adding new systems	Unchecked	Checked

Filter for EMC Smarts IP adapter

When you import topology through one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide*, use an open filter and select all system types as shown in Table 7-7.

Table 7-7 Filter for importing topology

Discovery filter field	Value
IP Address Range	*
System Name	*
System Description	*
SystemOID	*
System Types	Select all types

Filtering by system OID

To limit discovered systems by vendor, you can filter by the enterprise OID. The filter shown in Table 7-8 only matches systems of type Switch with an OID of 1.3.6.1.4.9, which is the enterprise OID for systems made by Cisco Systems, Inc.

Table 7-8 Filtering by OID

Discovery filter field	Value
IP Address Range	*
System Name	*
System Description	*
SystemOID	.1.3.6.1.4.1.9.*
System Types	Switch

How to create a discovery filter

For autodiscovery and topology import, discovery filters determine which systems are discovered and added to the topology. You create a filter by specifying a combination of characters and wildcards and selecting the types of systems to which the filter is applied. Appendix C, "Wildcard Patterns," discusses wildcard syntax.

To create a discovery filter:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Discovery Filters** tab.
4. Specify values for each of the discovery filter fields:
   - IP Address Range
   - System Name
   - System Description
   - SystemOID
   - System Types

   You must specify a value for each field. The System Name and System Description fields are case-sensitive.
5. Decide whether newly discovered systems matching this filter should be added to the topology or to the Pending Devices list. "Filtering with manual accept mode" on page 7-8 provides information about automatically or manually accepting new systems.
6. Optional: Check the **View Extended Filter** checkbox to view the extended options of the filter. The extended options enable you to specify the SNMP port, the SNMP version, the access mode, and the read community string or strings to be used when polling the systems that match an IP address within the IP Address Range of this filter.
7. Click **Add**, then **Apply**.

When you create multiple filters, systems are compared against the filters in the order in which the filters are arranged. When a system matches a filter, it is added to the discovery queue. Because of this behavior, put your more limiting filters first. You can arrange the order of the filters by selecting a filter and clicking the up or down arrows under the list of filters.

8

Understanding Discovery Results

This chapter describes IP Manager discovery errors and the Pending Devices list. It consists of the following sections:

- Discovery results 8-2
- Discovery error scenarios and results 8-3
- Discovery errors 8-5
- DiscoveryError notifications 8-9
- Pending Devices list 8-12
- Fine tuning parameter settings in discovery.conf 8-15

Discovery results

When the IP Manager attempts to discover a candidate system, one of the following occurs:

- Discovery successfully completes and the system and its components are added to the modeled topology.
- Discovery starts but communication is lost between the IP Manager and the system.
- Discovery starts but the system does not respond to the discovery request.
- Discovery starts but one of the discovery probes encounters an error during the discovery process.
- Discovery starts but cannot complete because of an insufficient number of volume licenses.
- Discovery is unable to start or successfully complete.

Successful discovery

When a candidate system is successfully discovered, the IP Manager creates an object in its repository for the system and for each of the system's components.

Unsuccessful discovery

If the discovery of a candidate system does not complete, the IP Manager places the system on the Pending Devices list. "Pending Devices list" on page 8-12 provides information about the Pending Devices list and the Discovery Progress window in which it appears.

Unsuccessful rediscovery

If the rediscovery of a system—that is, the discovery of a system that is already in the repository—does not complete, the IP Manager creates a DiscoveryError notification for the system, and that notification appears in the Notification Log Console view of the Global Console.

Discovery error scenarios and results

Table 8-1 describes different discovery error scenarios and results. Initial discovery refers to an attempt to discover a system that has not been previously discovered—not included in the modeled topology. Rediscovery refers to an attempt to discover a system that is already included in the topology.

Table 8-1 Discovery error scenarios and results

Discovery or rediscovery	Condition	Result
Discovery (initial discovery)	Cannot resolve hostname to IP address.	Add system to Pending Devices list. See "Discovering a system hostname that does not resolve to an IP address" on page 8-5.
	No response from ICMP ping.	For a seed system (manual discovery), add system to the Pending Devices list as UNDISCOVERED. For a system found through autodiscovery or topology import, discard system.
	No response from SNMP agent.	For a seed system (manual discovery), add system to the Pending Devices list as UNDISCOVERED. For a system found through autodiscovery or topology import, and ShowPendingNONSNMP set to TRUE, add system to the Pending Devices list as UNDISCOVERED. For a system found through autodiscovery or topology import, and ShowPendingNONSNMP set to FALSE, discard system. Also, see "System does not support SNMP" on page 8-7.
	ICMP response and SNMP response; current system limit exceeded.	For a seed system (manual discovery) or a system found through autodiscovery or topology import, display System limit exceeded message in the Discovery Progress window and add system to the Pending Devices list as UNDISCOVERED.
	ICMP response and SNMP response; manual accept mode enabled for discovery filter.	For a system matching the discovery filter, add system to the Pending Devices list as THROTTLED.
	Insufficient number of volume licenses, preventing complete discovery of systems.	Stop discovery process and send an OutOfLicense notification. Display "Out of license" message in the Discovery Progress window. Add any new discovery candidate systems to the Pending Devices list as UNDISCOVERED.
	Discovery completes successfully.	Classify system according to its sysObjectID and add to topology. If the OID is not recognized, classify system as a Node.

Table 8-1 Discovery error scenarios and results (continued)

Discovery or rediscovery	Condition	Result
Rediscovery	Receive no response from SNMP agent: SNMP request times out.	Send notification, typically Agent NotResponding, based on the correlation analysis, and send a DiscoveryError notification.
	SNMP agent encounters a loop, preventing the system from being discovered.	Classify system according to its sysObjectID, add to topology, and send a DiscoveryError notification.
	No response from ICMP ping.	Send notification, typically Down, based on the correlation analysis, and send a DiscoveryError notification.
	Discovery not able to contact SNMP agent on known address.	Send a DiscoveryError notification.
	SNMP agent fails authentication.	Send a DiscoveryError notification.
	The getExplains and getExplainedBy explanation modes are not identical.	Send a DiscoveryError notification. In general, as explained in "Explanation modes differ" on page 8-11, this notification is not a problem.

Discovery errors

An error occurring during the discovery of a candidate system results in the system being placed on the Pending Devices list. Each system entry on the Pending Devices list has an accompanying comment that describes the discovery error.

In most cases, the IP Manager writes discovery errors to its log file. The log file is named <IP Manager instance name>.log (for example, INCHARGE-AM.log) and is located in the BASEDIR/smarts/local/logs directory in the IP Management Suite installation area.

In some cases, the IP Manager prints a message, in the Discovery Progress window just below the Discovery Status section, that explains the discovery error.

The following is a representative—but not complete—list of discovery errors:

- Discovering a system hostname that does not resolve to an IP address
- SNMP agent violates SNMP protocol
- System does not support SNMP
- Incorrect read community string
- Insufficient number of volume licenses

Discovering a system hostname that does not resolve to an IP address

For a discovery candidate system identified by its hostname, the hostname might resolve to multiple IP addresses. If the IP Manager cannot reach the system using one of the first several IP addresses that the name resolution service—provided by the local operating system—finds, the IP Manager does not discover the system. Instead, the system is placed on the Pending Devices list, and subsequent attempts to discover it will also fail.

Discovery cannot resolve the hostname for a system when one of the following is true:

- The name resolution service returns more IP addresses than the operating system allows.
- None of the IP addresses returned by the operating system result in the IP Manager reaching the system's SNMP agent.

If you are using a seed file, you can solve the hostname resolution problem by simply replacing a system's hostname with an accessible IP address of the system's SNMP agent. If making this replacement is impractical because your seed file is generated automatically or because you are importing topology from an external source, read the following discussions of name resolution sources appropraite for your platform.

UNIX name resolution sources

All supported UNIX systems return up to 35 addresses for a given name. To avoid the name resolution problem, you must configure your system so that at least one usable IP address is among the 35 addresses returned.

UNIX systems translate names using one of three sources: DNS, NIS or NIS+, or a hosts file. The sources supported on a given system are specified by a host configuration file.

The most controllable source is the hosts file, /etc/hosts. All UNIX implementations that EMC Corporation personnel have tested will return (up to) the first 35 translations in order of appearance in the file.

DNS translations are more difficult to control. First, they are often updated automatically. Second, DNS servers automatically sort translations based both on internally defined and externally specified criteria.

EMC Corporation personnel have not been able to determine with certainty how translations made through NIS or NIS+ are ordered.

Windows name resolution sources

Windows systems provide up to six different sources for specifying name translations. Windows has different limits on the number of translations that the operating system will return, based on the source. Windows systems attempt to translate names from the following six sources:

- HOSTS file - Only one translation, the first one to appear in the file is returned.
- DNS — Multiple translations can be returned. EMC Corporation personnel have not been able to determine if there is a limit, and if so, what it is.
- WINS — Only one translation is returned; it is not clear how that translation is determined.
- Local broadcast — This source is not generally useful in IP-based networks.
- LMHOSTS file — The treatment of this source of translations appears to be equivalent to that of the HOSTS file.
- NetBIOS names — Windows-based hosts or Windows-based router cards might be named using their NetBIOS names when a DNS lookup fails. To avoid naming confusion, it is recommended that you disable NetBIOS over TCP/IP on Windows-based servers where the IP Manager is running. If it is not possible to disable NetBIOS over TCP/IP, then NetBIOS names should be consistent with DNS names.

If you run the IP Manager on a Windows system, you should avoid discovering systems using names with large numbers of translations. If unavoidable, and you are unable to discover the system, create a single entry for the system in the HOSTS file that specifies the IP address for the SNMP agent.

SNMP agent violates SNMP protocol

The IP Manager implementation of the SNMP protocol is strict in that it only sends or receives SNMP messages that conform to the SNMP standard. Unfortunately, not all SNMP implementations are as strict. The result is that the IP Manager programs might receive non-conforming SNMP messages. In some cases, the IP Manager can successfully interpret and handle non-conforming messages.

One such case is when an SNMP agent returns the values of an SNMP request in the wrong order. For example, the IP Manager requests the values for var1, var2, and var3, and the SNMP agent returns var1, var3, and var2. The IP Manager writes a message to its log file:

```
SWFE-W-Agent violates lexicographic ordering in response to GET_NEXT:
   Agent:172.16.1.94, First OID:
     .1.3.6.1.2.1.17.4.3.1.2.0.48.171.12.15.223
```

Although this problem does not hinder discovery, you might wish to contact the system vendor to see if a fix is available for the vendor's implementation of the SNMP protocol.

Note: By default, the IP Manager is configured to accept additional non-conforming SNMP messages because, in the runcmd_env.sh file, the value of the SM_SNMP_BUG_COMPATIBLE environment variable is set to TRUE. For information about environment variables, see the *EMC Smarts System Administration Guide*.

System does not support SNMP

Certain systems, such as desktop computers or printers, do not usually support SNMP. The discovery process assumes that a system supports SNMP. If a system does not support SNMP, the system is either not discovered or is placed on the Pending Devices list with a description that the SNMP request failed.

By default, systems found using autodiscovery that do not support SNMP are removed from the discovery queue. Systems imported from a seed file, or through the Add Agent command, that do not support SNMP are always added to the Pending Devices list.

If you wish to include autodiscovered, non-SNMP systems in the modeled topology, set the value of the ShowPendingNONSNMP parameter to TRUE in the discovery.conf file.

Note: Note that when autodiscovery is enabled, setting ShowPendingNONSNMP to TRUE can result in a large number of non-SNMP systems being added to the Pending Devices list.

When the value of ShowPendingNONSNMP is set to TRUE, candidate systems that do not support SNMP are placed on the Pending Devices list. "Managing systems on the Pending Devices list" on page 8-14 provides information about managing systems on the Pending Devices list.

Incorrect read community string

For errors concerning an incorrect read community string, you should examine the read community string and correct it if necessary. In the Notification Log Console, you can determine what community string is being used by double-clicking the notification and viewing the Details tab in the Notification Properties dialog.

- For manual discovery, a read community string can be specified in the seed file or in the Add Agent dialog box for each seed system, or one or more read community strings can be specified in the Topology tab of the Domain Manager Administration Console. "Adding and removing read community strings" on page 11-2 provides information about how to change community strings.
- For autodiscovery, one or more read community strings can be specified in a discovery filter or in the Topology tab of the Domain Manager Administration Console.

After correcting the read community string, add the system to the topology with the Add Agent command. You do not have to restart autodiscovery or re-import the entire seed file just to discover this one system.

Insufficient number of volume licenses

A license file for the EMC Smarts IP Management Suite may contain three different types of licensing: application licensing, feature licensing, and volume licensing. Volume licensing requires a license for each discovered system. Volume licensing

applies to IP Availability Manager but not to IP Performance Manager or IP Discovery Manager.

IP Availability Manager will retrieve blocks of volume licenses from the license server as needed when discovering systems. If IP Availability Manager discovers more systems than it is licensed to discover, it will generate the OutOfLicense notification to indicate that no more volume licenses are available and that no additional systems will be discovered.

When the OutOfLicense is notified, IP Availability Manager will complete the discovery of any system and its components in the midst of being discovered, including all virtual routers configured for a router or switch that implements virtual routers. When a router or switch implementing virtual routers is discovered, the chassis-based router/switch is counted as a system, and each virtual router is counted as a system. Any new discovery candidate systems are added to the Pending Devices list in the UNDISCOVERED state, and the message "Out of license" appears below the Discovery Status section of the Discovery Progress window.

If you receive an OutOfLicense notification, contact EMC Corporation Professional Services to acquire a new license file having a greater number of volume licenses. The *EMC Smarts IP Management Suite Installation Guide* provides the procedure to install a license file, and the *EMC Smarts System Administration Guide* provides more information about license management.

DiscoveryError notifications

Discovery errors are generated for systems that support SNMP. When an error is encountered during the rediscovery of such a system, the IP Manager notifies a DiscoveryError. If the AccessMode for a system is ICMPONLY and the system is not reachable by ICMP during discovery, the IP Manager does not notify a DiscoveryError.

To obtain more information about a DiscoveryError notification, attach the Domain Manager Administration Console to the Global Manager and double-click the notification in the Notification Log Console view to open the Notification Properties dialog. Select the Details tab to see the DiscoveryErrorInfo attribute. A Details tab is displayed for the IP Manager that notified the event.

The following is a list of errors that result in a DiscoveryError notification:

- SNMP request times out
- SNMP agent loops
- System Down
- Qualified access address not found
- System previously discovered fails authentication
- Explanation modes differ

SNMP request times out

A common error is that an SNMP request times out before it completes. There are a number of reasons why this error might occur.

- The system might be busy and therefore drops SNMP packets.
- The network might be busy and therefore drops or delays SNMP packets.
- A large bridge forwarding table might be causing the SNMP agent not to respond in time. Discovery uses the bridge forwarding table to determine connectivity.

If the error occurs because the system or network is busy, you should rediscover the system when the system or network is less busy.

If you notice an excessive number of timeouts during the discovery process, you should consider increasing the timeout values for discovery polling. "Fine tuning parameter settings in discovery.conf" on page 8-15 points you to the appropriate procedure.

SNMP agent loops

If a system has an SNMP agent with an improper MIB implementation, SNMP requests from the discovery process might loop over a table. Looping over a table is a violation of the standard protocol and prevents the IP Manager from retrieving the necessary information to fully discover a system.

When the IP Manager encounters such a problem, it notifies a DiscoveryError.

You can identify the location of the agent loop by invoking the following command from the BASEDIR/smarts/bin directory:

```
sm_snmpwalk <Host or IP address>
```

You should see an error message similar to the following:

```
SNMP-E-Agent 198.49.114.168, port 161 loops in response to
Get-Next/Bulk(.1.3.6.1.4.1.9.9.84.1.2.1.1.3.1.1.0.176.208.
   108.29.255)
```

Discovery continues with the next probe until the discovery process completes. However, if the error is the result of a problem with the system's MIB, you should contact the system vendor to see if an update is available.

System Down

In some cases, discovery fails because the system is Down (unreachable). When this situation occurs, the IP Manager notifies a DiscoveryError and lists the DiscoveryError as an impact of the Down problem. The value of the DiscoveryErrorInfo attribute is No response from ping.

Qualified access address not found

This error occurs during rediscovery of a system when no IP address object exists in the topology for an SNMP agent or each of the IP addresses associated with the SNMP agent is:

- An unmanaged IP address, or
- An IP address within the range of the ipExcludeList parameter in the discovery.conf file, or
- An IP address associated with an interface that is either in BACKUP or ONDEMAND mode.

Because it has no valid IP address to access the SNMP agent, the IP Manager notifies a DiscoveryError.

System previously discovered fails authentication

If, during a rediscovery, the SNMP agent of a system fails authentication, the IP Manager notifies a DiscoveryError. In the Notification Log Console, you can determine the reason for the error by double-clicking the DiscoveryError notification and viewing the Details tab in the Notification Properties dialog. Or, in the Topology Browser Console, you can determine the reason for the error by double-clicking the name of the failed system and viewing the Attributes tab in the system's properties dialog.

For an SNMPv1 or v2c agent, the most likely reason for this error is that the read community string for the agent has changed. "Incorrect read community string" on page 8-7 provides information about errors concerning an incorrect read community string.

For an SNMPv3 agent, the most likely reason for this error is that the authentication password, authentication protocol, or username for the agent has changed. "Seed file format" on page 6-3 provides information about SNMPv3 options.

Explanation modes differ

During the discovery process, the following error message might occur:

```
DX-W-MODES_DIFFER-Codebook==NO_EXPLANATION,
   alternate==RECURSIVE_WITHOUT_PROBLEM, so getExplains/getExplainedBy
   will trace each mode separately.
```

This message can be ignored unless you are running EMC Smarts Application Insight.

The default behavior of IP discovery is to trace each getExplains and getExplainedBy explanation mode separately. However, EMC Smarts Application Insight needs to trace with both modes combined, which is done only if the explanation modes are identical. If they are different, as is the default, EMC Smarts Application Insight will not work. Thus, a warning message is issued anytime getExplains or getExplainedBy is called in an Adapter Scripting Language (ASL) script.

Pending Devices list

The Pending Devices list, which appears in the Discovery Progress window, lists systems that are waiting to be discovered by the IP Manager. The IP Manager attempts to discover the systems on the Pending Devices list during the next pending discovery interval or when a pending discovery is manually invoked. ("Configure the pending discovery interval" on page 10-5 provides information about the pending discovery interval and pending discovery.)

To view the Pending Devices list, attach the Domain Manager Administration Console to the IP Manager and select Topology > Show Discovery Progress. Figure 8-1 is an example of a Pending Devices list.

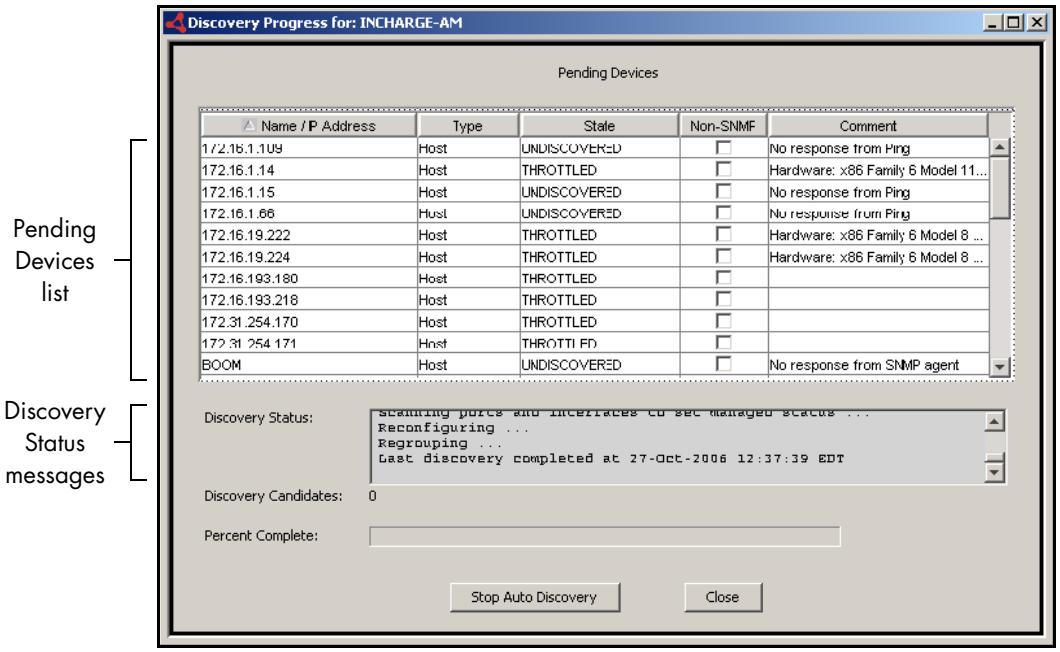


Figure 8-1 Pending Devices list containing multiple pending systems

A system is added to the Pending Devices list when one of the following occurs:

- The system does not respond to ICMP polls during a discovery.
- The system does not respond to SNMP polls during a discovery.
- The system is processed by autodiscovery and the manual accept mode is enabled. "Using manual accept mode" on page 4-3 provides information about manual accept mode.
- The number of discovered systems exceeds the current system limit. "Setting a current system limit for the topology" on page 4-4 provides information about setting the current system limit.
- The number of discovered systems exceeds the number of volume licenses available to the IP Manager. "Insufficient number of volume licenses" on page 8-7 provides information about volume licenses and the OutOfLicense notification.
- The system does not support SNMP, meaning that the system responded to ICMP polls but that the discovery process could not contact an SNMP agent for the system. "System does not support SNMP" on page 8-7 provides information about systems that do not support SNMP.

- The IP Manager receives an SNMP trap indicating that the configuration of the system might have changed. A system is placed on the Pending Devices list when one of the following traps is received:
  - coldStart (RFC 1215 MIB)
  - warmStart (RFC 1215 MIB)
  - a3ComSysModuleCardInsertEvent (3COM CB9000-MIB)
  - cefcFRUInserted (CISCO-ENTITY-FRU-CONTROL-MIB)
  - cefcFRURemoved (CISCO-ENTITY-FRU-CONTROL-MIB)
  - ciscoConfigManEvent (CISCO-CONFIG-MAN-MIB)
  - sysConfigChangeTrap (CISCO-STACK-MIB)
- The topology is autodiscovered, or imported through one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide*, and the ShowPendingNONSNMP parameter in the discovery.conf file is set to TRUE.

Information provided by Pending Devices list

The Pending Devices list contains five columns that provide the following information for each system:

- Hostname or IP address of the system that could not be discovered.
- Type, which is one of the system types listed in "Unitary computer systems and management agents" on page A-10.
- State is either UNDISCOVERED or THROTTLED, as described in Table 8-2 and in "Using manual accept mode" on page 4-3.
- Non-SNMP is a checkbox for indicating that the system does not support SNMP.
- Comment describes why the system is on the Pending Devices list, or contains the system's SysDescr value of the system if the current system limit has been exceeded.

Table 8-2 describes the discovery states and the actions you can take to resolve the situation.

Table 8-2 Discovery states

Discovery state	Description	Available actions
UNDISCOVERED	Indicates that the system was not successfully discovered. Another attempt to discover the system will occur when the Pending Devices list is next processed.	• Designate the system as one that does not support SNMP. The system will be re-classified as a Host during the next pending discovery Interval. • Rediscover the system. • Remove the system.
THROTTLED	Indicates that the system was autodiscovered with manual accept mode enabled and placed on the Pending Devices list.	• Accept the system and it will be discovered at the next full discovery, pending discovery, or rediscovery. • Remove the system.

Managing systems on the Pending Devices list

Depending on a system's state, you can invoke one of the following four actions on a system on the Pending Devices list. Right-click the system to see the available actions.

- Accept pertains to systems in the THROTTLED state, which means that the system matched an autodiscovery filter configured for manual accept mode. If you accept the system, the IP Manager changes the state of the system to UNDISCOVERED.
- Remove is available for all systems on the Pending Devices list. Selecting this action removes the system from the Pending Devices list. If you remove a system, the IP Manager prompts you to confirm the removal.
- Mark Non-SNMP is available for all systems on the Pending Devices list. When you designate a system as Non-SNMP, the IP Manager reclassifies the system as a Host and adds it to the topology at the next discovery interval assuming that the system responds to the IP Manager ICMP request. Systems marked as non-SNMP are treated similarly to systems marked as ICMPONLY.

  Once a system has been marked Non-SNMP in the Pending Devices list, you cannot change its status. You must rediscover the system manually by using the Add Agent command or by importing the system from a seed file.
- Rediscover initiates a discovery of the system.

Fine tuning parameter settings in discovery.conf

The discovery.conf file contains configuration parameters that you can edit to improve performance during topology discovery. Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes these parameters and provides instructions on modifying them.

The chapter also includes recommendations for setting the ICMP and SNMP polling configurations. You should find these recommendations helpful.

9

Customizing Discovery

This chapter describes how to customize the IP Manager discovery process. It consists of the following sections:

- Certifying non-network systems..... 9-2
- Using discovery hook scripts 9-4

Certifying non-network systems

After your network has been discovered, you might want to keep certain systems such as personal computers or printers in your topology. Because these are non-network systems, they are classified as GENERIC by the IP Manager discovery process. You can, however, add such systems to a field certification file so that the IP Manager properly recognizes these systems.

For EMC Smarts version 4.0.1 or later, you should add non-network systems to the field certification file, oid2type_Field.conf. User-made changes to this file will not be overwritten by subsequent SmartPacks. This file is located in the BASEDIR/smarts/conf/discovery directory.

Editing oid2Type_Field.conf

To certify non-network systems:

1. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the *oid2Type_Field.conf* file.

   ```
   sm_edit conf/discovery/oid2Type_Field.conf
   ```

2. Observe the syntax of system entry in "Syntax of oid2type_Field.conf" and add the systems that you wish to certify.
3. Save and close the file. The modified version of the file is saved to the *BASEDIR/smarts/local/conf/discovery* directory.
4. Restart the IP Manager to make your changes take effect.

Syntax of oid2type_Field.conf

The following example illustrates the syntax of a system entry in the oid2type_Field.conf file. Each entry contains the sysObjectID of the system followed by several fields that describe the system. You can use a pound sign (#) to add comments to the file.

```
# Structure of system entry
<sysObjID> {
    TYPE = Host
    VENDOR = <manufacturer>
    MODEL = [<model_name>]
    CERTIFICATION = TEMPLATE

INSTRUMENTATION:
    Interface-Fault = MIB2
    Interface-Performance = MIB2
}
```

Table 9-1 describes each field.

Table 9-1 Field descriptions for oid2type_Field.conf

Field	Description
TYPE	Type of system. This value should always be Host, regardless of the actual system type.
VENDOR	Manufacturer of system. This value must be a single word.
MODEL	Optional field that if present usually contains the model name of the system. This field can be more than one word. Note that the field name is required.

Table 9-1 Field descriptions for oid2type_Field.conf (continued)

Field	Description
CERTIFICATION	Level of certification for system. This value should always be TEMPLATE. Template indicates that the system is recognized by the IP Manager but that the discovery of the system has not been tested by EMC Corporation personnel.
Interface-Fault	MIB used for analysis. This value should always be MIB2.
Interface-Performance	MIB used for analysis. This value should always be MIB2.

The following example shows the type of certification information added to certify a host and a printer. Note that both systems are classified as hosts.

```
# Dell Personal Computer
.1.3.6.1.4.1.311.1.1.3.1.1 {
    TYPE = Host
    VENDOR = Dell
    MODEL = Latitude
    CERTIFICATION = TEMPLATE

INSTRUMENTATION:
    Interface-Fault = MIB2
    Interface-Performance = MIB2
}

# This is an example of a printer (TYPE = Host)
# IBM Network Printer 24
.1.3.6.1.4.1.2.3.6 {
     TYPE = Host
     VENDOR = IBM
     MODEL = Network Printer 24
     CERTIFICATION = TEMPLATE

INSTRUMENTATION:
     Interface-Fault = MIB2
     Interface-Performance = MIB2
}
```

Using discovery hook scripts

In addition to the post-processing steps performed automatically during the fourth phase of the discovery process, you can use discovery hook scripts to specify additional post-processing steps to occur at various stages of post-processing. (For example, you could use a post-processing script to manage or unmanage IP objects during discovery.) The discovery hook scripts are written in the EMC Smarts proprietary Adapter Scripting Language (ASL).

When the discovery hook scripts run

The discovery hook scripts are identified in the DISCOVERY_CUSTOM.import file located in the BASEDIR/smarts/conf/discovery directory of the IP Management Suite installation area.

Each time the IP Manager performs a discovery (an initial discovery, a full discovery, a pending discovery), it runs the main discovery probes plus the discovery hook scripts (custom probes) identified in the DISCOVERY_CUSTOM.import file. By default, running the discovery hook scripts has no effect on the discovery process because the scripts are just template files, meaning that you must open the scripts with an editor and add the code that performs the actual processing.

Note: You must have administrative privileges to edit the discovery hook scripts.

Discovery hook script files

The IP Manager provides five discovery hook script files, all of which are located in the BASEDIR/smarts/rules/discovery/custom directory of the IP Management Suite installation area. They are:

- custom-start-fulldscv.asl
- custom-start-system.asl
- custom-end-system.asl
- custom-start-post.asl
- custom-end-post.asl

The ASL scripts are invoked at the following points of the discovery process:

- The custom-start-fulldscv.asl script runs before a full discovery.
- The custom-start-system.asl script runs before initial system discovery.
- The custom-end-system.asl script runs after initial system discovery and before discovery post-processing.
- The custom-start-post.asl script runs before discovery post-processing.
- The custom-end-post.asl script runs after discovery post-processing.

Modifying a discovery hook script file

You use the sm_edit utility to edit these scripts to add the ASL convenience patterns, parsing rules, and processing rules that perform the actual processing. You can also add additional initialization code and additional EOF or DEFAULT code.

To modify an ASL file:

1. Go to the *BASEDIR/smarts/bin* directory in the IP Management Suite installation area and issue the following command to open the ASL file. For example:

   ```
   sm_edit rules/discovery/custom/custom-start-system.asl
   ```

2. Add your ASL code.
3. Save and close the file. The modified version of the file is saved to the *BASEDIR/smarts/local/rules/discovery/custom* directory.
4. Restart the IP Manager to make your changes take effect.

Example discovery hook script file

The following example shows how the custom-start-system.asl and custom-end-system.asl scripts identify systems for processing. The discovery process automatically assigns the name of the SNMP agent hosted by the system currently being discovered to the AgentName variable. The system scripts use the value of AgentName to obtain the name of the system by invoking the getsystem() operation on the SNMP agent object.

```
/*
* Passed in argument
*/
default AgentName = "";

/*
* To obtain SNMPAgent object
*/
agentObj = object();
agentObj = object("SNMPAgent", AgentName)?IGNORE;
if (agentObj->isNull()) {
   stop();
}

/*
* To obtain system object
*/
systemObj = agentObj->getSystem();
if (systemObj->isNull()) {
stop();
}

/*******************************************************/

START {
/*
* Add Parsing rules here if needed.
*/
.. eol
} do {
/*
* Add processing rules here.
*/
stop();
}
```

10

Scheduling and Manually Invoking Discovery

This chapter explains how to schedule full and pending discovery and how to manually invoke discovery for the IP Manager. It consists of the following sections.

- Automatic and manual discovery 10-2
- Discovery considerations 10-3
- Scheduling automatic discovery 10-4
- Invoking discovery from the console 10-6
- Invoking discovery from the command line 10-7

Automatic and manual discovery

After the initial discovery by the IP Manager, you can schedule automatic discovery at a regular interval and manually invoke discovery at any time. You perform these tasks through a Domain Manager Administration Console attached to the IP Manager, or from the command line using the dmctl utility.

For both automatic and manual discovery, the following two methods of discovery are available:

- Full discovery
- Pending discovery

Manual rediscovery of individual systems is also available.

Full discovery

During the initial full discovery, as explained in Chapter 4, "Preparing and Initiating Discovery,", the IP Manager populates its repository with the topology discovered in the managed network environment. During any subsequent full discovery, the IP Manager attempts to rediscover the topology in its repository.

Rediscovering the topology reveals any changes in the configuration or connectivity of the managed systems and keeps the topology current. For example, if a card is added to a switch, the IP Manager must rediscover the switch so that it can add the new card to the topology. And if new systems have been added to the network and autodiscovery is enabled, the IP Manager will discover the new systems during the rediscovery.

Pending discovery

During a pending discovery, the IP Manager attempts to discover the systems on the Pending Devices list. The Pending Devices list contains the names of the systems that could not be discovered by the IP Manager.

You can view the Pending Devices list at any time by selecting Topology > Show Discovery Progress from the Domain Manager Administration Console. "Pending Devices list" on page 8-12 describes the various aspects of the Pending Devices list.

Discovery considerations

Initiating a full discovery for a large topology can take a considerable amount of time, ranging from one hour to two or more hours, depending on the actual size of the topology. Keep this in mind when scheduling full discovery for the IP Manager.

Also, consider scheduling automatic full discovery during off hours (5:00 in the morning, for example). The *EMC Smarts IP Management Suite Deployment Guide* recommends scheduling full discover at least weekly and pending discovery at least daily. Typically, the duration of a pending discovery is much shorter than the duration of a full discovery.

Running the discovery more often will provide a more accurate network topology, but will also cost more in terms of network and system resources. The IP Manager continues event-processing and analysis during discovery but runs at a somewhat slower rate.

Scheduling automatic discovery

You can schedule full discovery and pending discovery intervals:

- Scheduling a full discovery interval determines how often the IP Manager attempts to rediscover the existing systems in its repository.
- Scheduling a pending discovery interval determines how often the IP Manager attempts to discover the systems on the Pending Devices list.

By default, the full discovery interval is disabled, and the pending discovery interval is enabled and set to six hours. The parameters that control scheduled discovery are located on the Topology tab page of the Domain Manager Administration Console.

The full discovery interval should not be set to less than 60 minutes, and the Pending discovery interval should not be set to less than five minutes. Violating either of these restrictions will result in the following exception being thrown from the Domain Manager Administration Console:

```
Apply changes failed: CI-E-ICF-TopologyManager:
   reprobePeriod:constraint MinimumProbePeriod MR-Violation of hard
   constraint
```

Configure the full discovery interval

When scheduled full discovery is enabled, the full discovery interval is counted from the time when the IP Manager is started. So, if the full discovery interval is set to 1 day and the IP Manager is started at 5:00 AM, the full discovery will occur each day at 5:00 AM.

The IP Manager's topology manager program starts a full discovery every full discovery interval unless the previous full discovery is still in progress, in which case the topology manager ignores the discovery request and writes an exception to the IP Manager's log file. For example, if the full discovery interval is set to 1 hour and the actual full discovery takes 1.3 hours to complete, the topology manager will ignore the discovery request every other discovery interval, meaning that the actual full discovery will occur every 2 hours.

Note: This behavior is different for pre-6.3 topology managers for which the cycle time is the summation of the discovery time and the discovery interval:
|<- discovery time ->|<- discovery interval ->|<- discovery time ->| ...

To enable scheduled full discovery and to specify a full discovery interval:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Click the **Enable Full Discovery** checkbox.
5. Type a value in the **Interval** field for full discovery and select a time interval (**Seconds**, **Minutes**, **Hours**, or **Days**) from the drop-down list. Ensure that the interval is equal to or greater than 60 minutes.
6. Click **Apply**.

Configure the pending discovery interval

The pending discovery interval may be configured to a different value than the full discovery interval.

The pending discovery interval is counted from the time when the IP Manager is started. So, if the pending discovery interval is set to six hours (the default) and the IP Manager is started at 5:00 AM, the pending discovery will occur every six hours thereafter, that is, at 11:00 AM, at 5:00 PM, at 11:00 PM, and so on.

To specify a pending discovery interval other than six hours:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Type a value in the **Interval** field for pending discovery and select a time interval (**Seconds**, **Minutes**, **Hours**, or **Days**) from the drop-down list. Ensure that the interval is equal to or greater than five minutes.
5. Click **Apply**.

Invoking discovery from the console

From the Domain Manager Administration Console, you can use any of the following methods to manually invoke a discovery:

- To discover all managed systems, including those on the Pending Devices list, select **Topology > Discover All**.
- To discover all systems on the Pending Devices list, select **Topology > Discover Pending**.
- To discover a single system on the Pending Devices list, right-click the system on the Pending Devices list and then select **Rediscover** from the pop-up menu.
- To rediscover a single system in the topology tree, click the system and then select **Topology > Rediscover**.

  You can also rediscover a system by right-clicking it and selecting **Rediscover** from the pop-up menu.

Invoking discovery from the command line

From the command line, you can invoke a full discovery or a pending discovery using the dmctl utility. Doing so is useful if you want to schedule a full or pending discovery at a specific time using batch processing or the EMC Smarts sm_sched utility.

To use the dmctl utility to initiate a full discovery, go to the BASEDIR/smarts/bin directory in the IP Management Suite installation area and invoke the following command on one line:

```
dmctl -s <IP manager instance name> [-b <broker>] invoke
     ICF_TopologyManager::ICF-TopologyManager discoverAll
```

For example:

```
dmctl -s INCHARGE-AM invoke
     ICF_TopologyManager::ICF-TopologyManager discoverAll
```

Note: The option -b <broker> is used to specify the host and port (<host:port>) location of the Broker if the IP Manager is attached to a Broker other than "localhost:426."

To use the dmctl utility to initiate a pending discovery, go to the BASEDIR/smarts/bin directory in the IP Management Suite installation area and invoke the following command on one line:

```
dmctl -s <IP Manager instance name> [-b <broker>] invoke
     ICF_TopologyManager::ICF-TopologyManager discoverPending
```

When prompted for a username and password, specify an administration account; for example, *admin* and *changeme*. The *EMC Smarts System Administration Guide* provides more information about the dmctl utility.

11

Working with the Topology

This chapter describes additional configuration tasks related to topology and discovery. It consists of the following sections:

- Adding and removing read community strings 11-2
- Managing and unmanaging topology objects 11-4
- Extracting a seed file from the IP Manager 11-8
- Removing systems from the topology 11-9
- Saving the topology 11-10

Adding and removing read community strings

You can specify the read community strings used by discovery in several different places:

- Discovery filter
- Seed file
- Add Agent command

If one or more read community string are specified for a discovery filter, those community strings are used when polling the systems that match an IP address within the IP address range of the discovery filter. Chapter 7, "Creating Discovery Filters," provides information about specifying read community strings for a filter.

If no read community string is specified in a discovery filter, or no community string is specified in the seed file or through the Add Agent command, or the specified community string is not correct, the IP Manager chooses a community string in the following manner:

- Tries any read community string or strings listed in the Read Community Strings window (Topology tab) of the Domain Manager Administration Console.
- If there are no community strings in the Read Community Strings window, reads the default read community string in the discovery.conf file.
- If no read community strings are found for a system, places the system on the Pending Devices list (during an initial discovery attempt) or notifies a DiscoveryError (during a rediscovery attempt).

Note: Initial discovery versus rediscovery is discussed in Chapter 8, "Understanding Discovery Results."

How to add read community strings

To add a read community string to the Read Community Strings window:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Type a read community string in the **Community Strings** field.
5. Click **Add**. By default, you can list up to four read community strings in the **Read Community Strings** window.
6. Click **Apply**.

The changes that you make to the Read Community Strings window are applied to systems discovered after the changes are made. Systems already discovered are not affected.

How to delete read community strings

To delete a read community string from the Read Community Strings window:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, click the IP Manager name to display a multiple-tab window in the right panel of the console.
3. Select the **Topology** tab.
4. Select the community string to be removed in the **Read Community Strings** window.
5. Click **Remove**, then **Apply**.

The default read community string public is automatically included in the Read Community Strings window. If you do not want the IP Manager to use the public community string, you must remove public here and remove public from the defaultReadCommunity parameter in the discovery.conf file.

Note: Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the configuration parameters in the discovery.conf file and provides instructions on modifying the parameters.

How to specify additional read community strings

To specify additional read community strings, you need to change the value of the MaximumCommunities parameter in the discovery.conf file.

Managing and unmanaging topology objects

A managed object refers to an object whose IsManaged attribute has a value of TRUE, and an unmanaged object refers to an object whose IsManaged attribute has a value of FALSE. A managed object is monitored using ICMP and SNMP—unless a different access mode is specified—to determine the object's status and connectivity. An unmanaged object is not polled or rediscovered but remains present in the modeled topology.

The Manage and Unmanage commands enable you to control whether an object is monitored by the IP Manager. Unmanaging a system might be useful when, for example, a switch or a card is taken offline for maintenance. Unmanaging duplicate (overlapping) IP objects (that correspond to duplicate IP addresses discovered using the IP tagging feature) is practical for several reasons:

- An overlapping IP address is typically the address on a system's physical interface, not the address of the system's SNMP agent. Thus, an overlapping IP address is typically not "reachable."
- If a pair of overlapping IP addresses are both reachable and managed by the IP Manager, the IP Manager will not be able to determine which of the systems bound to those IP addresses is responding to the ICMP or SNMP polls. If one of the overlapping IP address is reachable and the other is not, managing the reachable one and unmanaging the unreachable one might make sense.

Note: The Manage and Unmanage commands are recursive. For example, if you unmanage a switch, all of the ports and cards that belong to the switch are unmanaged.

There are three methods you can use to control the managed status of objects.

- You can set the managed status of individual objects using the Manage and Unmanage commands. "Managing and unmanaging individual objects" on page 11-6 provides information regarding these commands.
- You can use the Interface Management Policy provided with IP Availability Manager and IP Performance Manager to set the managed status for interfaces that belong to System Resource Groups. The Interface Management Policy, accessible through the Polling and Thresholds Console, enables you to set a system-level policy to unmanage interfaces. By default, this policy is not active and all interfaces in the topology are managed. Chapter 7, "Customizing Polled Data Thresholds," in the *EMC Smarts IP Management Suite Configuration Guide* describes the Interface Management Policy and provides instructions on changing its setting.
- You can create IP tag filter groups to not only tag specific IP addresses on a discovered system but to also set those IP addresses to managed or unmanaged when the system is discovered. Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* describes the IP tagging feature and provides instructions on creating IP tag filter groups.

Note: If you are using one of the EMC Smarts IP adapters described in the *EMC Smarts IP Adapters User's Guide*, you must allow the third-party source to control the managed status of network objects. When the adapter's topology reader synchronizes the topology of the IP Manager with the topology of the third-party source, the topology reader must be able to overwrite the managed status of objects in the IP Manager's topology.

Determining the managed status of an object

The value of the IsManaged attribute shows the managed status of an object. IsManaged is a Boolean attribute: TRUE when an object is managed, and FALSE when the object is unmanaged.

To check the value of an object's IsManaged attribute:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, select the object to display the object's properties window in the right panel of the console.
3. In the object's properties window, click the **Attributes** tab and locate the **IsManaged** attribute. The value of this attribute indicates the managed status of the object.

Types of objects that can be managed or unmanaged

You can set the managed or unmanaged status for the following types of objects:

- Systems such as routers, switches, and hosts
- Physical packages such as cards
- Network adapters such as ports and interfaces
- Protocol endpoints such as IP interfaces
- Management agents such as SNMP agents
- Logical devices such as fans, temperature and voltage sensors, and processors

The ability to unmanage IP interfaces enables you to develop flexible management policies. When you unmanage an IP interface, the underlying physical interface continues to be managed because the status of the IP interface is determined using ICMP, and the status of the physical interface is determined using SNMP.

You can unmanage the IP interfaces on systems where ICMP pings are not allowed because, for example, of a firewall. When you unmanage these IP interfaces, the IP Manager no longer polls them using ICMP. More importantly, however, is that the IP Manager still manages the underlying physical interfaces through SNMP.

Rules governing manage and unmanage

The Manage and Unmanage operations are recursive. When you unmanage an object, all the management agent services and other objects that are functionally dependent on that object become unmanaged.

The following list of rules determines whether an object can be managed or unmanaged and explains the recursive nature of the Manage and Unmanage commands.

- When you unmanage a system, a management agent service that is hosted by or is a part of that system also becomes unmanaged. For example, a system such as a router hosts an SNMP agent service. If you unmanage a router that hosts an SNMP agent service, the SNMP agent service is also unmanaged.

- When you unmanage a system, a logical device that is a part of that system becomes unmanaged. Logical devices include objects such as fans, temperature and voltage sensors, and processors. Ports and interfaces also become unmanaged. If you unmanage a system such as a router, the interfaces and sensors that are part of that router are not managed nor can they be managed while the parent object is unmanaged. Likewise, if you manage a system, all of its components are managed except those that are explicitly unmanaged.
- When you add a logical device or service to an unmanaged system, that logical device or service automatically becomes unmanaged. For example, if an interface is added to an unmanaged host, the new interface is also unmanaged.
- When you unmanage a service, the protocol endpoint that is associated with that service becomes unmanaged.
- When you unmanage a logical device or a protocol endpoint, any other protocol endpoint that is layered over that logical device or protocol endpoint becomes unmanaged. For example, if you unmanage an interface, the subinterfaces or IP addresses layered over that interface are also unmanaged.
- When you unmanage a card, the managed state of any sub-cards, network adapters realized by the card, or systems packaged by the card remain managed. As such, monitoring and analysis of these items continues.

Managing and unmanaging individual objects

To manage or unmanage an object using the Manage and Unmanage commands:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, select the object to be managed or unmanaged, right-click, and then select **Manage** or **Unmanage** from the pop-up menu.
3. Select **Topology > Reconfigure**.

Note: When you manually change the management state of an entity, either through the Domain Manager Administration Console or through the command line, you must reconfigure. Until you reconfigure, the change will not take effect.

Persistent management state

IP Availability Manager and IP Performance Manager retain the management state of interfaces, ports, IP addresses and SNMP agents in the following situations:

- After a system has been recertified
- If there has been a change to the DNS configuration
- If the name format has been changed, as described in "How discovery names a system" on page 3-4

Duplicate IP address errors

When two systems with the same IP address (overlapping IP addresses) are discovered, the IP Manager does not create an IP object for the second occurrence of the same IP address; instead, the IP Manager creates a DuplicateIP object representing the duplicate IP addresses and deletes the IP object for the first occurrence. The IP Manager behaves this way even for overlapping IPs that are tagged, using the IP tagging feature explained in "Discovery and IP tagging" on page 1-8 and "Name Resolution probe" on page 2-13, if the resulting tagged IP names are identical.

Note: To clarify, if an IP address has already been discovered on one system and the same IP address is then found to be configured as the only SNMP agent address on another system, the IP Manager will proceed as follows: not discover the second system, remove the first system from its repository, create a DuplicateIP object, and generate a DuplicateIP notification. Of course, if the IP objects created for the duplicate IP addresses are tagged properly, the IP Manager will discover both systems.

For a duplicate IP address error, the IP Manager initiates the following actions:

- Classifies the IP address as a DuplicateIP
- Disables all other IP-related notifications for the IP address
- Generates a notification regarding the duplicate IP address

To correct a duplicate IP address error:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, select the DuplicateIP object, right-click, and then select **Delete** from the pop-up menu.
3. Use the IP tagging feature to give each of the duplicate IP addresses a unique tag.

   Consult Chapter 9, "Creating IP Tagging Filters," in the *EMC Smarts IP Management Suite Configuration Guide* to complete this step.
4. Rediscover all systems that contain duplicate IP addresses.

   Consult "Invoking discovery from the console" on page 10-6 to complete this step.

If you choose not to use IP tagging to correct the problem, you can clear the DuplicateIP notification by unmanaging the duplicate IP address. To do so, select the DuplicateIP object, right-click, and then select Unmanage from the pop-up menu.

Extracting a seed file from the IP Manager

You can retrieve a list of the IP addresses and read community strings of the SNMP agents managed by the IP Manager.

To obtain a list of SNMP agents, invoke the sm_tpmgr utility on the host where the Domain Manager is running. This utility produces a text file named seedfile.log that lists the IP address of the SNMP agents in one column and their corresponding read community strings in a second column. For SNMPv3 agents, of course, no read community strings appear in the second column.

To obtain a list of SNMP agents and their read community strings, invoke the following command from the BASEDIR/smarts/bin directory:

```
sm_tpmgr -s INCHARGE-AM [-b <broker>] --dump-agents --output=seedfile
```

The -s argument specifies the name of the IP Manager. The -b argument is optional and specifies the host and port (<host:port>) location of the Broker if the IP Manager is attached to a Broker other than "localhost:426." The --dump-agents argument extracts the information regarding the SNMP agents from the IP Manager. The --output=seedfile argument creates a file named seedfile.log in the BASEDIR/smarts/local/logs directory. More information regarding the sm_tpmgr utility is available with the --help argument.

Removing systems from the topology

To remove a system from the topology:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. In the left panel of the Domain Manager Administration Console, select the system to be deleted, right-click, and then select **Delete** from the pop-up menu.

When you remove a system, all of its components and connections are also removed.

Saving the topology

Every six hours, the IP Manager automatically saves important topology and configuration information to a repository file in the BASEDIR/smarts/local/repos/icf directory. The configuration information pertains to the groups and settings applied to the managed systems. When the IP Manager is restarted, it loads the saved information from the repository file.

The name of the repository file is taken from the name of the IP Manager. For example, if INCHARGE-AM is the name of an instance of IP Availability Manager, its repository file would be INCHARGE-AM.rps.

To manually save the topology to the repository file:

1. Attach the Domain Manager Administration Console to the target IP Manager. ("Procedure for opening the Domain Manager Administration Console" on page 5-2 provides instructions on opening this console.)
2. On the Domain Manager Administration Console, select **Topology > Save Topology**.

A

ICIM Network Concepts

This appendix describes the EMC Smarts management model and the hierarchy of topology classes used by the IP Manager to represent the managed network objects. It consists of the following sections:

- ICIM overview A-2
- ICIM models chart A-3
- Two ICIM hierarchies A-4
- More information about object types A-10
- Topology examples A-16
- ICIM instrumentation classes A-19

ICIM overview

ICIM is the EMC Smarts management model, and is based on the industry-standard Common Information Model defined by the Distributed Management Task Force. ICIM uses an object-oriented design to present a conceptual view of the managed environment. The object-oriented design promotes the use of inheritance, relationships, abstraction, and encapsulation to improve the quality and consistency of management data.

ICIM is an information model that captures notions applicable to all areas of management. It is a set of object classes, common characteristics and features (properties), relationships (associations), and behavior (methods) that provides a basic vocabulary for describing and analyzing managed objects. The object classes represent network equipment, computer equipment, and application services.

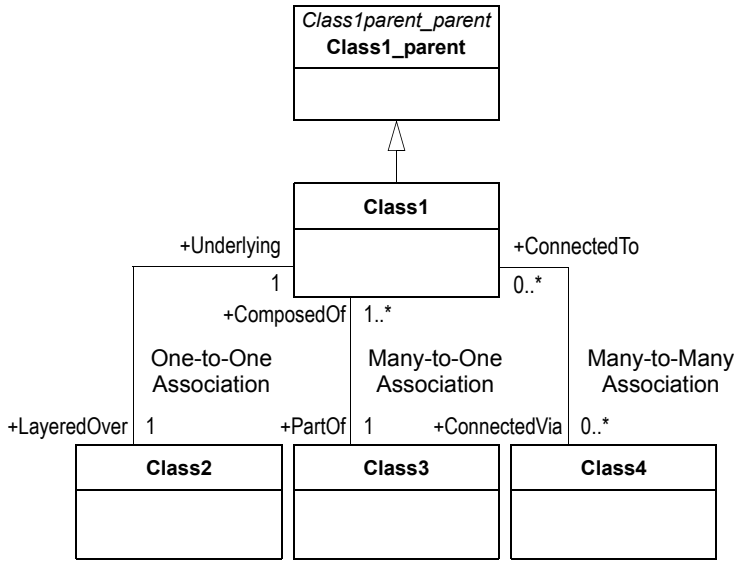
The ICIM classes, or models, are written in MODEL, which is an object-oriented programming language that defines the types of entities that an EMC Smarts application will manage, the potential relationships between instances of these entities, the problems to be diagnosed, and the impacts (symptoms) caused by these problems.

The goal of EMC Corporation is to use ICIM to represent all aspects of the managed environment, not just a single problem domain. To this end, EMC Corporation has created various ICIM classes to address network, system, and application problem domains. The ICIM classes used by the IP Manager, for example, address the transport level of the IP network problem domain.

ICIM models chart

The ICIM network and system classes used by the IP Manager to create representations of Layer 2 (data-link) and Layer 3 (network) equipment and their connections are illustrated in the *EMC Smarts Common Information Model Infrastructure Models Chart*. The information presented in Table A-1 should prove helpful in understanding the class definitions and notations on this chart.

Table A-1 How to read inheritance and relationships In the ICIM models charts

Notation for ...	Examples
Inheritance: • Hollow-tipped arrow points to parent class. • Italicized name (if present) in upper-right corner of class's box is class's parent class. (ICIM depend upon class hierarchy and inheritance in order to pass attributes and other characteristics to other classes.)	 Class1 is LayeredOver Class2; Class2 is Underlying Class1. Class1 is PartOf Class3; Class3 is ComposedOf one or more Class1's.
Relationships: • One-to-one association • Many-to-one association • Many-to-many association (Generally, in ICIM, for every relationship, there is an inverse relationship. For example, the relationship PartOf is the inverse relationship of ComposedOf.)	

Note: The *EMC Smarts ICIM Reference* provides descriptions of ICIM classes, relationships, and connections.

Two ICIM hierarchies

An essential feature of ICIM is that it describes the managed network in terms of two parallel, but distinct, hierarchies: physical and logical. The former describes the real-world components that make up your managed system. A physical component is something to which you could attach an inventory tag.

A logical component, on the other hand, is visible through network protocols, provides some network service, or connects logical components. It is created through the operation of hardware and software—you cannot touch it.

The physical hierarchy

The physical hierarchy consists of subclasses of the class ICIM_PhysicalElement, which describe any objects of a system that have distinct physical identities. All current ICIM physical objects are actually drawn from subclasses of PhysicalPackage, which represents physical objects that can contain or host other physical objects.

The most common class in the physical hierarchy is Card. Cards are used to model any object capable of carrying signals or providing a mounting point for other physical components. Other classes you will see include Chassis, which represents the physical objects that enclose other objects and provide some definable functionality, and Rack, which represents an enclosure in which Chassis are placed.

The most basic relationship in the physical hierarchy is ComposedOf, which relates a physical object to its component parts (which in turn are PartOf the enclosing object). A Chassis is ComposedOf the Cards placed in it; a Card used as a motherboard might in turn be ComposedOf Cards plugged into it; and so on.

Table A-1 identifies the physical package type classes that can be instantiated by the IP Manager.

Table A-1 Physical package type classes instantiated by the IP Manager

Class	Available to	Comment
Card	All IP Managers	
Chassis	All IP Managers	
NASChassis	IP Availability Manager Extension for NAS	Represents the chassis of a Celerra NAS gateway. Requires a feature license; also available to all other IP Managers when the feature license is installed.
Rack	All IP Managers	

Note: The indentations in the Class column indicate class hierarchy.

The logical hierarchy

The logical hierarchy consists of subclasses of the class ICIM_LogicalElement. These describe abstract system objects, ranging from UnitaryComputerSystems and ProtocolEndpoints to RedundancyGroups. Logical objects are creations of software; they exist only as long as the software implementing their capabilities continues to run.

Logical objects can be divided into the following categories of object types:

- Unitary computer systems
- Network adapters
- Device connections
- Service access points
- Logical links
- Collections
- Management agents
- Logical devices
- Redundancy groups

Unitary computer systems

A unitary computer system, or just system, represents a computer-based network entity. Important classes of the computer system include the Host (a workstation or server) and the following relay systems: Hub, Bridge, Switch, and Router. The relay systems forward packets at Layer 1 (a Hub), Layer 2 (a Bridge or Switch), or Layer 3 (a Router) in a network.

Table A-2 identifies the unitary computer system type classes that can be instantiated by the IP Manager.

Table A-2 Unitary computer system type classes instantiated by the IP Manager

Class	Available to	Comment
Bridge	All IP Managers	
Host	All IP Managers	
FileServer	IP Availability Manager Extension for NAS	Represents a Data Mover component of a Celerra NAS gateway. Requires a feature license; also available to all other IP Managers when the feature license is installed.
Hub	All IP Managers	
LoadBalancer	All IP Managers	
Node	All IP Managers	Represents a discovered system having an unrecognized OID. The IP Manager uses standard MIB-II data to analyze Node objects.
Probe	All IP Managers	
Router	All IP Managers	
Firewall	All IP Managers	
MSFC	All IP Managers	Represents a Multilayer Switch Feature Card.
RSFC	All IP Managers	Represents a Router Switch Feature Card.
RSM	All IP Managers	Represents a Router Switch Module.

Table A-2 Unitary computer system type classes instantiated by the IP Manager (continued)

Class	Available to	Comment
Switch	All IP Managers	
TerminalServer	All IP Managers	
Uncertified	All IP Managers	Represents a discovered system that has yet to be certified by EMC Corporation. The IP Manager does not monitor Uncertified objects.

Note: The indentations in the Class column indicate class hierarchy.

Network adapters

A network adapter represents a Layer 2 connection point. Two classes are common: Interfaces, such as Ethernet interfaces that connect routers to an Ethernet; and Ports, which connect Interfaces to Layer 2 systems. An Ethernet connection to an Ethernet Switch is an example of a Port.

Table A-3 identifies the network adapter type classes that can be instantiated by the IP Manager.

Table A-3 Network adapter type classes instantiated by the IP Manager

Class	Available to	Comment
Interface	All IP Managers	
Port	All IP Managers	

Device connections

A device connection represents the path between a pair of network adapters. The most important examples are Cables, TrunkCables, and NetworkConnections.

ConnectedVia plays a role for device connections that is similar to its role for logical links. A network relay system such as a Switch can be connected via a Cable to another relay system such as a Router, or via a TrunkCable to another relay system such as a Switch. Two Routers joined by a virtual circuit whose intermediate systems are not included in the topology (for example, a Frame Relay PVC) are connected by a NetworkConnection.

Table A-4 identifies the device connection type classes that can be instantiated by the IP Manager.

Table A-4 Device connection type classes instantiated by the IP Manager

Class	Available to	Comment
NetworkConnection	All IP Managers	
Cable	All IP Managers	
TrunkCable	All IP Managers	

Note: The indentations in the Class column indicate class hierarchy.

Service access points

A service access point describes a logical endpoint that can be used to gain access to some network service. Important service access points are IP, which represents the point at which an IP address can be reached; and Media Access Control (MAC), which similarly represents the point at which a Layer 2 address can be reached. Service access points are often described as protocol endpoints.

A service access point is related to some lower level network object by the LayeredOver relationship. Thus, an IP address would be LayeredOver an Interface.

Table A-5 identifies the service access point type classes that can be instantiated by the IP Manager.

Table A-5 Service access point type classes instantiated by the IP Manager

Class	Available to	Comment
DLCI	All IP Managers	Represents a Frame Relay Data Link Connection Identifier endpoint.
HSRPEndpoint	All IP Managers	Represents a Hot Standby Router Protocol endpoint. Requires a feature license.
IP	All IP Managers	Represents an IPv4 address endpoint.
DuplicateIP	All IP Managers	Represents two (or more) overlapping IPv4 addresses.
VPNIP	All IP Managers	Represents a VRF IP address endpoint.
DuplicateVPNIP	All IP Managers	Represents two (or more) overlapping VRF IP addresses.
IPv6	IP Availability Manager Extension for IPv6	Represents an IPv6 address endpoint. Requires a feature license; also available to all other IP Managers when the feature license is installed.
DuplicateIPv6		Represents two (or more) overlapping IPv6 addresses.
MAC	All IP Managers	Represents a Media Access Control endpoint.
STPNode	All IP Managers	Represents a Spanning Tree Protocol node.
TCP	Not instantiated	
VirtualTCP	All IP Managers	Represents a virtual Transmission Control Protocol endpoint.

Note: The indentations in the Class column indicate class hierarchy.

Logical links

A logical link models the path that connects service access points to each other. The most common logical link class is IPNetwork, which represents an IP (sub)network. ConnectedVia describes the relationship between a protocol endpoint and a logical link. Two IP addresses can communicate directly when they are ConnectedVia the same IPNetwork. A connection between two IP addresses that are not on the same IPNetwork must go through one or more Routers.

Table A-6 identifies the logical link type classes that can be instantiated by the IP Manager.

Table A-6 Logical link type classes instantiated by the IP Manager

Class	Available to	Comment
IPNetwork	All IP Managers	Represents a Layer 3 logical link between IP objects.
VPNIPNetwork	All IP Managers	
IPv6Network	IP Availability Manager Extension for IPv6	Requires a feature license; also available to all other IP Managers when the feature license is installed.
VLAN	All IP Managers	Represents a Layer 2 logical link between MAC objects.

Note: The indentations in the Class column indicate class hierarchy.

Collections

A collection represents a group of managed objects, such as an isolated region of inter-connected network devices.

Table A-7 identifies the collection type classes that can be instantiated by the IP Manager.

Table A-7 Collection type classes instantiated by the IP Manager

Class	Available to	Comment
Partition	All IP Managers	

Management agents

A management agent is a logical object that contains the information necessary to represent, configure, or manage some functionality provided by a system or software feature. Management agents include SNMP agents, which are applications that provide monitoring and other management functions for a computer system.

Table A-8 identifies the management agent type classes that can be instantiated by the IP Manager.

Table A-8 Management agent type classes instantiated by the IP Manager

Class	Available to	Comment
SNMPAgent	All IP Managers	
VRAgent	All IP Managers	

Note: The indentations in the Class column indicate class hierarchy.

Logical devices

A logical device is an emulation of a hardware entity that might or might not be realized in physical hardware. Logical devices include power supplies and processors, sensors that measure temperature or voltage characteristics of a computer system, cooling devices such as fans, and storage media that store data and allow data retrieval.

Table A-9 identifies the logical device type classes that can be instantiated by the IP Manager.

Table A-9 Logical device type classes instantiated by the IP Manager

Class	Available to	Comment
Disk	IP Server Performance Manager	All logical device type classes except Disk and FileSystem are also available to: • IP Availability Manager and IP Performance Manager running as a single process • IP Discovery Manager and IP Performance Manager running as a single process **Note:** The service startup command created for IP Discovery Manager during the installation starts IP Discovery Manager and IP Performance Manager as a single process. A feature license is required for IP Performance Manager to function as an IP Server Performance Manager.
Fan	All but IP Availability Manager	
FileSystem	IP Server Performance Manager	
Memory	All but IP Availability Manager	
PowerSupply		
Processor		
TemperatureSensor		
VoltageSensor		

Redundancy groups

A redundancy group is a logical object that contains two or more elements that participate in a redundant configuration. ICIM supports redundancy groups of systems, cards, network adapters, network connections, and protocol endpoints.

Table A-10 identifies the redundancy group type classes that can be instantiated by the IP Manager.

Table A-10 Redundancy group type classes instantiated by the IP Manager

Class	Available to	Comment
CardRedundancyGroup	All IP Managers	
NetworkAdapterRedundancyGroup	All IP Managers	
NetworkConnectionRedundancyGroup	All IP Managers	
ProtocolEndpointRedundancyGroup	Not instantiated	
HSRPGroup	All IP Managers	Requires a feature license; also available to all other IP Managers when the feature license is installed.
SystemRedundancyGroup	All IP Managers	

Note: The indentations in the Class column indicate class hierarchy.

Joining the two hierarchies

The physical and logical hierarchies are not independent: The objects of the physical hierarchy exist exactly to provide implementations for logical objects that implement features of interest in the network. The Realizes relationship joins a physical object to the logical objects for which it acts as the physical embodiment or implementation.

More information about object types

This section provides additional details about the following categories of ICIM object types:

- Unitary computer systems and management agents
- Network adapters and device connections
- Chassis and cards
- Protocol endpoints and logical links
- Redundancy groups

Unitary computer systems and management agents

A unitary computer system contains cards and network adapters, and hosts protocol endpoints and management agents. As a logical entity it might represent the abstraction of a physical system such as a switch or a logical entity such as a virtual router within a Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) router.

The IP Manager discovers the following system objects:

- Bridge — A bridge is a protocol-independent system that connects two LAN segments.
- FileServer — A file server is a system that hosts file shares.
- Firewall — A firewall is a system that controls the flow of traffic between networks.
- Host — A host is a general purpose computer, such as a workstation or server.
- Hub — A hub is a system that connects multiple physical segments. Active hubs are multi-port repeaters, which means that they repeat signals received on any port to all the other ports.
- LoadBalancer — A load balancer is a system that distributes client traffic across multiple web servers.
- Multilayer Switch Feature Card (MSFC) — A card in a switch that performs routing between VLANs.
- Node — A node identifies a system that is monitored using generic network management instrumentation. Nodes are probed for standard MIB-II information but not for enterprise-specific information such as system resources.
- Probe — A probe is a system that monitors networks or other systems. An example is a Remote Monitoring (RMON) probe.
- Router — A router is a system or, in some cases, software in a computer that determines the next network point to which a packet should be forwarded as it travels toward its destination. A router is connected to at least two networks and decides which way to send each information packet based on its current understanding of the state of the networks to which the router is connected.

  A router might also be a virtual router, which is a software emulation of a router implemented within a physical router or switch. Each virtual router has its own independent IP routing and forwarding tables, which permit the same routing and forwarding of packets as with a standard router. Virtual routers are often used with VPNs to allow a greater separation of VPN traffic while using the same equipment.

- Router Switch Feature Card (RSFC) — A card in a Catalyst switch that runs Cisco IOS router software and is used to perform routing between VLANs.
- Router Switch Module (RSM) — An RSM is a router installed as a card in a switch to perform routing between VLANs.
- Switch — A switch is a system that switches packets, typically at wire speeds, between physically separate network segments.
- TerminalServer — A specialized system that connects terminals to a network.
- Uncertified — An uncertified object identifies a system that has yet to be certified by EMC Corporation.

A unitary computer system may be associated with the physical package, chassis or card, in which it is housed. An example is shown in Figure A-1 where a router is packaged in a chassis. The chassis packages a system, the router, and the router hosts a management agent and is composed of network adapters.

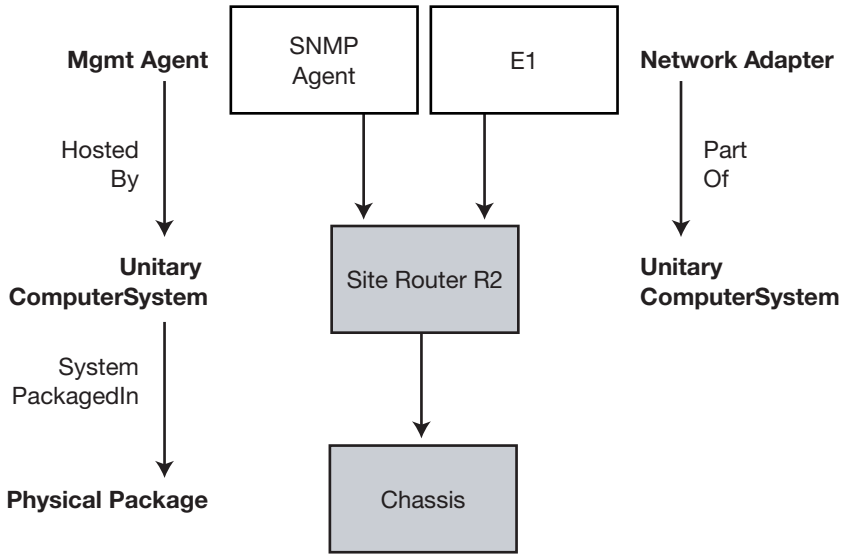


Figure A-1 Packaging of a system

Chassis and cards

A card is a physical object that plugs into a chassis. (See Figure A-2.) A chassis is also a physical object. Card and chassis are modeled as physical objects (rather than as logical abstractions of physical objects) because there are aspects of the physical manifestation that are useful to represent, namely asset information such as the serial number. A card is associated with one or more network adapters that are realized by the card. A card might also be part of another card as in the case of a daughter card.

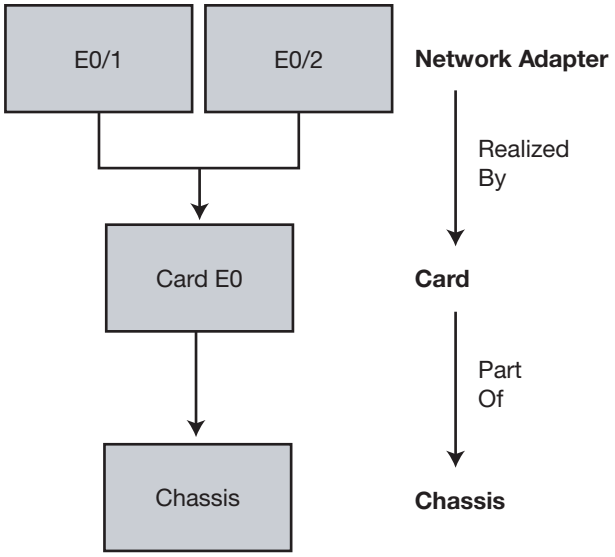


Figure A-2 Chassis, card, and network adapters

Network adapters and device connections

A network adapter represents a connection point such as an Ethernet interface on a host or an access port on a switch. Instances of the Port or Interface class can represent a physical interface or port, one with a connector to which you attach a cable, or a logical adapter such as a Frame Relay subinterface. In the case of a physical interface or port, the network adapter is a logical abstraction of the physical adapter.

The IP Manager discovers the following network adapter objects:

- Port – A port is where the physical connection to a network segment is made. A port may have a Media Access Control (MAC) address layered over it.
- Interface – An interface is where the physical connection to a network is made. An interface may have a MAC address, an IP address, or both layered over it.
- Subinterface – A subinterface is a logical division of a physical interface. A physical interface can be divided into one or more subinterfaces. For example, in a typical Frame Relay network, a physical interface is configured with multiple virtual circuits called Data Link Connection Identifier (DLCI) interfaces, and each virtual circuit is associated with a subinterface.

Note: Subinterface objects, similar to interfaces, are instances of the Interface class.

A network adapter may be layered over another network adapter. The layering represents (models) the dependency between network adapters at different protocol layers. For example, in Figure A-3, the Frame Relay subinterfaces S.1 and S.2 are layered over the physical interface S.

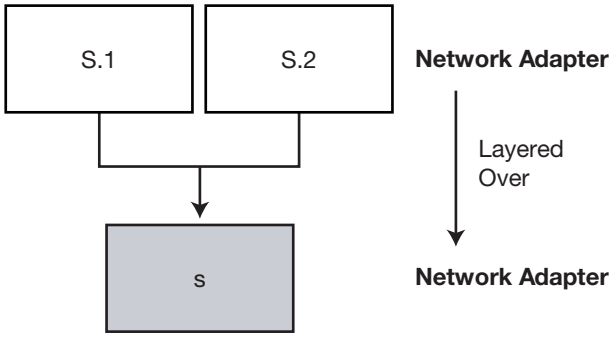


Figure A-3 Layering of network adapters

Another example of layered interfaces is a channelized DS1 interface where each channel is an interface, and each channel is layered over the DS1 interface.

A network adapter is also classified according to its intended use, indicated by the Mode attribute. The mode can be one of normal, on-demand, or backup. A value of normal describes an active and operational network adapter, such as those of type Ethernet or serial. An on-demand adapter becomes active only when it is needed. An example of such an adapter is a PPP or SLIP interface that hangs up when idle to reduce dialup charges. Such an adapter fluctuates between up and down under normal operating conditions. A backup adapter is one that activates only when there is a failure on a primary link. Backup adapters are inactive under normal operating conditions. An example of a backup adapter is an ISDN adapter where the physical interface remains up but the logical adapters layered over the physical interface are down until the link is activated.

A network adapter attaches to the network through a device connection. The connection may be the abstraction of a physical entity such as a cable, or it may represent a purely logical entity such as a virtual connection over a Wide Area Network. A device connection is typically attached to two network adapters, but ICIM does not preclude it from attaching to more. An example might be a point-to-multipoint virtual connection.

Note: The IP Manager discovers and monitors both point-to-point and point-to-multipoint Frame Relay virtual circuit connections.

The IP Manager discovers the following device connection objects:

- Cable – A cable is a connection between a port and an interface.
- TrunkCable – A trunk cable is a connection between two ports.
- NetworkConnection – A network connection is a connection between two interfaces. A network connection can be a logical connection or a physical connection. An example of a logical connection is when routers are connected via a virtual circuit and none of the intermediate network systems are included in the topology. An example of a physical connection is when routers are connected via a serial or point-to-point connection.

Network connections may exist at different network adapter layers. Consider the case of a Frame Relay interface on an access router, as shown in Figure A-4. The connection at interface S represents the physical connection to the Frame Relay access system, while the connection at subinterface S.2 is a virtual connection across the Wide Area Network to the peer subinterface at the remote site.

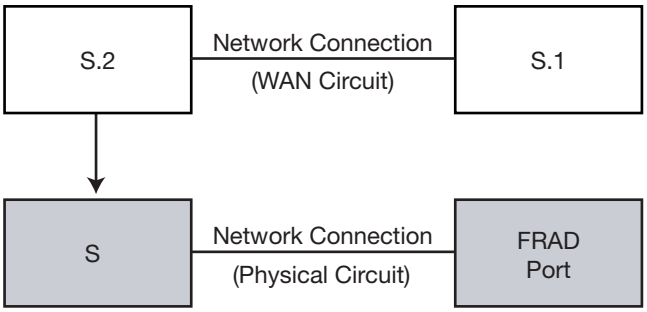


Figure A-4 Layering of network connections

Network connections may be layered over other network connections. This enables the discovery process to build a topological representation of the complex dependencies between connections at different layers. For example, the Wide Area Network circuit connection from Figure A-4 could be layered over the various logical and physical connections it traverses within the Wide Area Network.

Protocol endpoints and logical links

A protocol endpoint is a higher layer service access point such as an IP endpoint, a LAN endpoint, or a PVC endpoint. A protocol endpoint is distinguished by a layer-specific protocol address. An IP endpoint is identified by its IP address, a LAN endpoint by its MAC address, a PVC endpoint by its DLCI number. Protocol endpoints may be layered over network adapters or other protocol endpoints and hosted by systems. (See Figure A-5.) An IP endpoint on a LAN system will usually be layered over a network adapter and a MAC endpoint. This layering represents the dependency between the IP layer and the data-link layer and the dependency between the IP layer and the interface through which it is accessed.

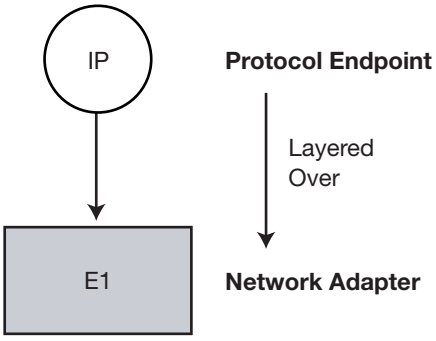


Figure A-5 IP Protocol endpoint layered over a network adapter

Protocol endpoints connect to logical links. A logical link is a purely logical connection at the layer of the protocol endpoint. An IP network is a layer 3 logical link, a VLAN is a layer 2 logical link, and a PVC is a layer 2 logical link connecting its DLCI endpoints. Logical links may be layered over other logical links or device connections.

Redundancy groups

A redundancy group is a logical entity that contains two or more objects that participate in a redundant configuration. For example, a remote site accessed through two routers may be modeled as a redundancy group that contains the two routers. If

one router fails then the redundancy is at risk. If both fail then all capability is lost. ICIM supports redundancy groups of network adapters, network connections, cards, and unitary computer systems.

The *EMC Smarts IP Availability Manager User's Guide* provides information about creating redundancy groups.

HSRP groups

An HSRP group is a redundancy group of routers using a proprietary protocol from Cisco called Hot Standby Routing Protocol. The group provides router backup in the event of a router system failure or an interface failure.

In an HSRP group, several routers connected to the same segment of an Ethernet, FDDI, or token-ring network work together to present the appearance of a single virtual router on the LAN. Because the routers share the same IP and MAC addresses, the hosts on the LAN are able to continue forwarding packets to a consistent IP and MAC address in the event of a router system failure. The process of transferring the routing responsibilities from one system to another is transparent to the user.

The *EMC Smarts IP Availability Manager User's Guide* provides information about creating HSRP groups.

Topology examples

The following topology examples combine some of the objects previously described and include the relationships between the different objects. All of these objects and relationships are created automatically by the discovery process.

Host connected to a switch

The first example, shown in Figure A-6, shows all the basic objects and relationships created to represent a connection between two objects: a single-homed host (H1) and a switch (S1).

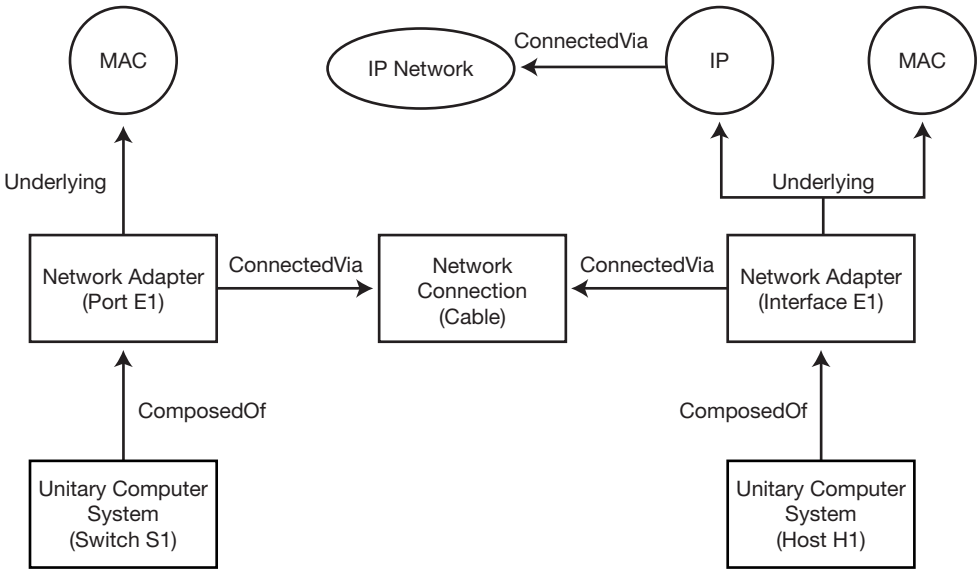


Figure A-6 Topology of a connection between a host and switch

The interface on the host and the port on the switch are both network adapters. Both have an associated MAC endpoint, and the interface has an associated IP endpoint logically connected to an IP subnet. The network connection between the two network adapters is modeled as a cable, which, in ICIM, represents a connection between an interface and a port.

Wide area network

The second example, shown in Figure A-7, shows a Wide Area Network connection between Hub Router H1 and Site Router R2. Typically, such a topology would include additional subinterfaces, which are not shown to simplify the diagram.

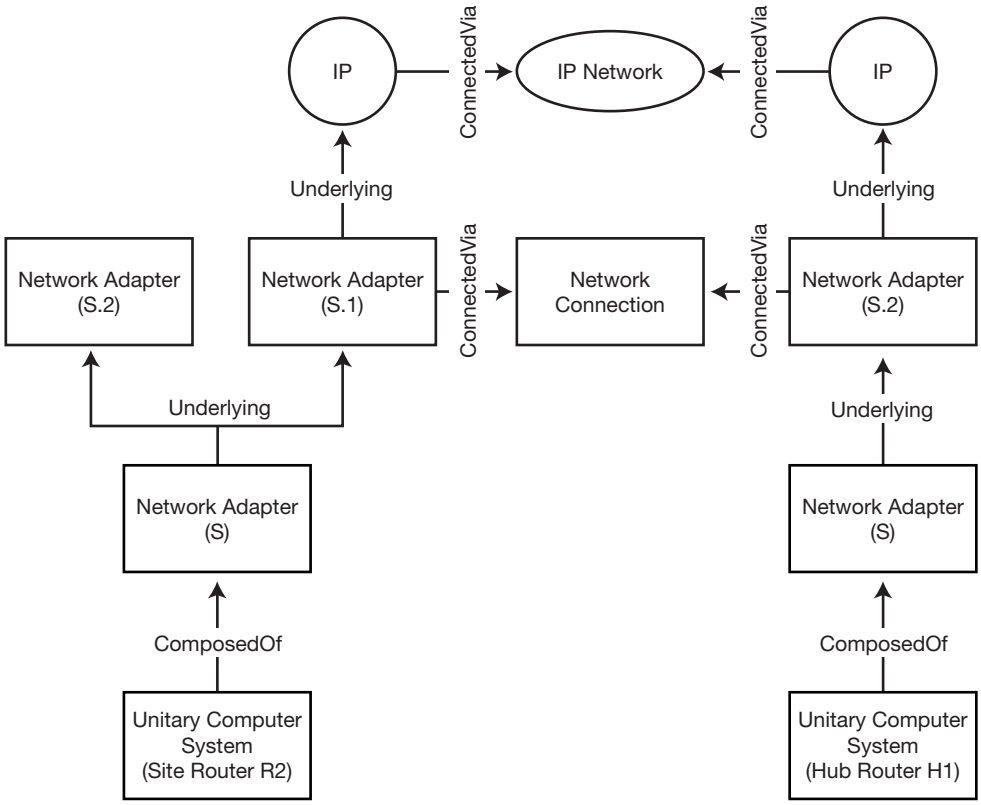


Figure A-7 Topology of a connection over a wide area network

This topology example illustrates the layering between the physical interfaces and the subinterfaces (S.1 and S.2). The subinterfaces have associated IP endpoints while the physical interface does not. This example also shows the connectivity at the different layers: the network connection represents a layer 2 circuit connection between subinterfaces, and the IP network represents the point-to-point IP subnet riding on the circuit. The IP layer would be omitted for an unnumbered connection.

Switch with routing module

The last example, shown in Figure A-8, shows a switch with a routing blade. It illustrates how ICIM represents (models) both the physical and logical objects of a network and creates relationships between them. The physical objects include the switch chassis and router switch module, while the logical objects include the switch, router, and logical interface.

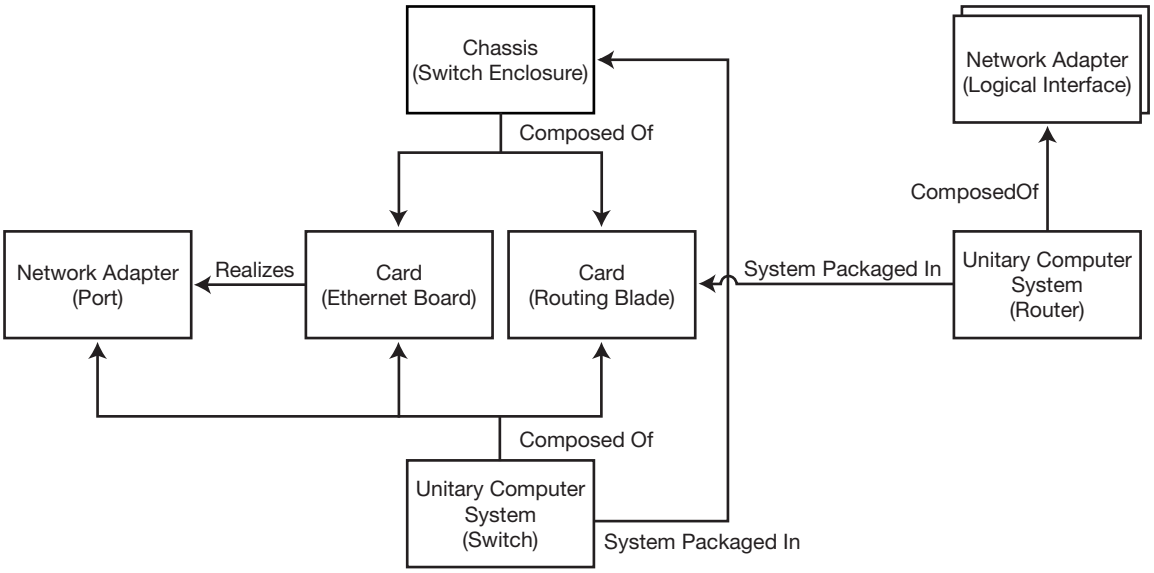


Figure A-8 Topology of physical and logical objects

ICIM instrumentation classes

The ICIM core class for instrumentation is ICIM_Instrumentation, and the ICIM core class for managed object is ICIM_ManagedElement. Figure A-9 illustrates the relationship between these classes.

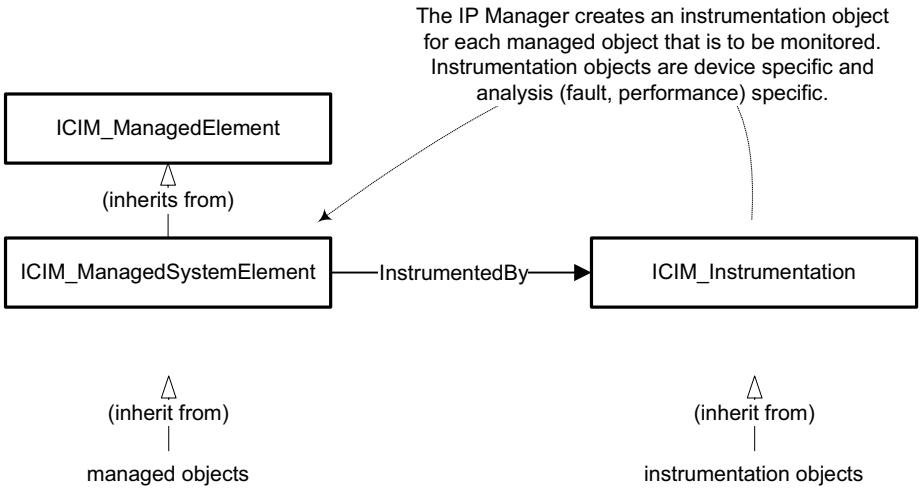


Figure A-9 Relating managed objects to instrumentation objects

The ICIM_Instrumentation class is the parent of all instrumentation classes used by the IP Manager or any other EMC Smarts Domain Manager, and the ICIM_ManagedElement class is the parent of all managed object classes used by the IP Manager or any other EMC Smarts Domain Manager.

For service-assurance applications like the IP Manager that use SNMP polling to monitor the discovered systems, instrumentation defines the methods to use to search the MIBs of a discovered system for system status information. IP Availability Manager instantiates instrumentation classes for the purpose of fault management, whereas IP Performance Manager and IP Server Performance Manager instantiate instrumentation classes for the purpose of performance management.

The oid2type configuration files in the BASEDIR/smarts/conf/discovery directory contain an INSTRUMENTATION block that specifies the instrumentation classes to use to monitor the components of a particular system (device). This section may contain any or all of the following parameters for a particular system:

- Environment
- CPU/Memory
- Chassis-Fault
- Chassis-Performance
- Card-Fault
- Interface-Fault
- Interface-Performance
- Interface-Ethernet-Performance
- Port-Fault
- Port-Performance
- Port-Ethernet-Performance

The value specified for the Environment parameter identifies the instrumentation class to use to monitor the system for Fan, PowerSupply, TemperatureSensor, VoltageSensor, and other types of health status. The value specified for the CPU/Memory parameter identifies the instrumentation class to use to monitor the device for Processor and Memory status. The remaining parameters identify the instrumentation classes for Chassis, Cards, Interfaces, and Ports.

To see the instrumentation (and the many other) classes instantiated by an IP Manager, go to the BASEDIR/smarts/bin directory in the IP Management Suite installation area where the IP Manager is running and issue the following command:

```
dmctl -s <IP Manager instance name> [-b <broker>] getc
```

Note: The option -b <broker> is used to specify the host and port (<host:port>) location of the Broker if the IP Manager is attached to a Broker other than "localhost:426."

Here are some examples of the instrumentation classes that you might see:

- CPU_Performance_Adtran
- Card_Fault_3COM_CB9000
- Chassis_Fault_CiscoStack
- Disk_Fault_BluecoatRsc
- Disk_Fault_HostResources
- EthernetAdapter_Performance
- Fan_Fault_FoundryEnvMon
- HSRPEndpoint_Fault_Cisco
- Interface_Fault_Cosine_VIf
- Interface_Performance_CiscoRouter
- Memory_Performance_Compaq_Physical
- NetworkAdapter_Fault_MIB2
- NetworkAdapter_PerformanceData_MIB2
- Port_Fault_LucentP550
- Port_Performance_dot3_Ethernet
- PowerSupply_Fault_Tellabs
- TemperatureSensor_Fault_JuniperEnvMon
- VRAgent_Fault_Juniper
- VoltageSensor_Fault_RedBackEnvMon

B

Polling Used During Discovery

This chapter describes the two types of polling performed by the IP Manager during discovery: Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP). It consists of the following sections:

- What is ICMP? B-2
- What is SNMP? B-3
- Controlling EMC Smarts ICMP and SNMP polling B-6

What is ICMP?

Internet Control Message Protocol is tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation. Some of ICMP's functions are to announce network errors, announce network congestion, announce timeouts, and to assist in troubleshooting network problems.

ICMP supports an Echo function that sends a packet on a round--trip between two hosts. Ping, which is a common network management tool based on the Echo function, transmits a series of packets to measure average round--trip times and to compute loss percentages. Ping is central to the IP Manager ICMP polling.

IPv4 ICMP poller

The standard ICMP poller for the IP Manager is an IPv4 ICMP poller, which sends ICMP messages for use with IP version 4. During the first phase of the discovery process, the IP Manager sends initial IPv4 ICMP requests and initial IPv4 SNMP requests to determine whether the system is reachable and supports SNMP. If the system is reachable and supports SNMP, the IP Manager:

- Discovers the system,
- Gathers IPv4 addresses for the system,
- Discovers the components of the system, and
- Uses IPv4 ICMP and IPv4 SNMP to monitor the system.

IPv6 ICMP poller

IP Availability Manager Extension for IPv6 uses the standard IPv4 ICMP poller and an IPv6 ICMP poller, which sends ICMP messages for use with IP version 6. During the first phase of the discovery process, IP Availability Manager Extension for IPv6 sends initial IPv4 ICMP requests and initial IPv4 SNMP requests to determine whether the system is reachable and supports SNMP. If the system is reachable and supports SNMP, IP Availability Manager Extension for IPv6:

- Discovers the system,
- Gathers IPv4 and IPv6 addresses for the system,
- Discovers the components of the system, and
- Uses IPv4 ICMP, IPv6 ICMP, and IPv4 SNMP to monitor the system.

IP Availability Manager Extension for IPv6 enhances the analysis of networks that include dual-stack systems: systems configured with both IPv4 and IPv6 addresses.

Note: All IP products support IPv4 SNMP polling, but none support IPv6 SNMP polling.

What is SNMP?

The Simple Network Management Protocol is used by network management applications for discovery, fault management, and performance management. SNMP is an application-layer protocol that exchanges management information between a network management application, such as IP Availability Manager, and one or more managed network objects, called agents.

SNMP, itself, does not provide network management. What SNMP does provide is a framework, or infrastructure, on which network management applications can be built.

SNMP key components

Figure B-1 illustrates the SNMP key components: management station, agent, management information base, and SNMP protocol.

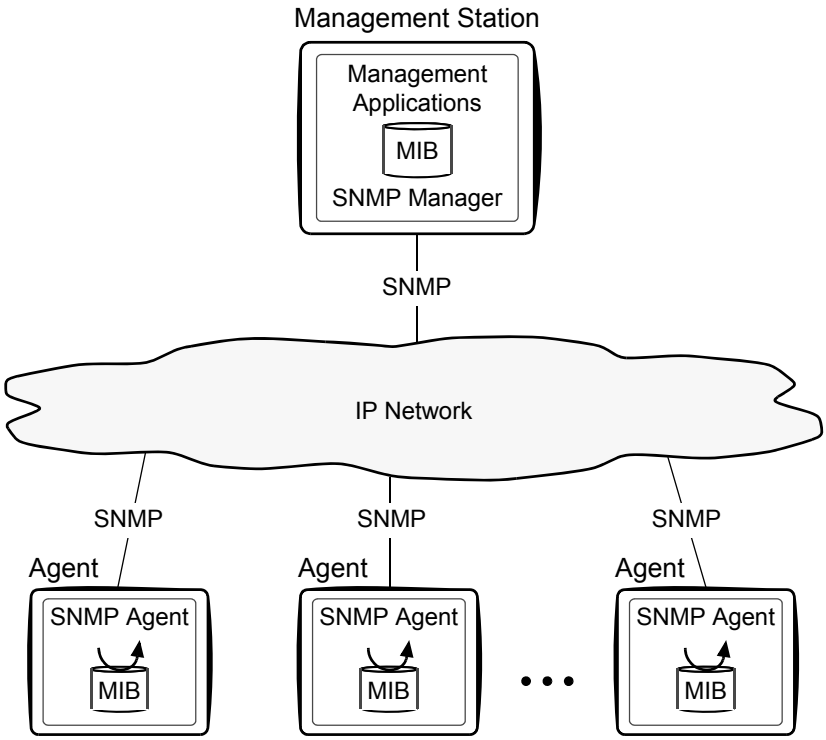


Figure B-1 SNMP-managed network

Each SNMP agent or SNMP manager maintains a MIB. An agent MIB contains information about the network system managed by the SNMP agent and about the agent itself. A manager MIB contains network management information extracted by the SNMP manager from each agent MIB.

SNMP basic operations

SNMP includes the following basic operations:

- GET: enables the SNMP manager to retrieve (read) the value of MIB objects at the SNMP agent.
- SET: enables the SNMP manager to set (write) the value of MIB objects at the SNMP agent.
- NOTIFY: enables an SNMP agent to asynchronously notify the SNMP manager of significant but unsolicited events (traps).
- INFORM: enables an SNMP manager to asynchronously send an alert (similar to a trap) to another SNMP manager.

SNMP versions

Currently, there are three versions of the SNMP protocol: v1, v2C, and v3. SNMPv1 and V2c offer rudimentary authentication and authorization schemes, whereas SNMPv3 offers robust authentication and authorization schemes plus data privacy.

SNMPv1 security

SNMPv1 security consists of a pairing of an SNMP agent with some arbitrary set of SNMP managers to form an SNMP community. Each SNMP community is given a name called the community name or community string for the community. Common community names are public (read-only), private (read-write), and trap. The permissions (read-only, read-write) for a community name indicate the read-write permissions of an SNMP manager when using that community name to access an agent's MIB. Community names can be thought of as passwords to SNMPv1 agents.

SNMPv2c security

Like SNMPv1, SNMPv2c uses the notion of communities to establish trust between SNMP managers and SNMP agents. Unlike SNMPv1, SNMPv2c uses an updated version of the structure of management information (SMI), which extends the MIB object tree, allows several new data types, and makes a number of other changes. SNMPv2c is more efficient than SNMPv1 and has better error-handling capabilities.

SNMPv3 security

SNMPv3 provides integrity, authenticity, data privacy, and access control for SNMP messages exchanged between an SNMP manager and the managed SNMP agents. Unlike the community-based administrative model of SNMPv1 and V2C, SNMPv3 unambiguously identifies the source and destination of each SNMP message. (An SNMPv3 manager or agent may have multiple party—user—identities.) And instead of using community names to establish trust between SNMP managers and SNMP agents, SNMPv3 uses the following security-related services to establish trust:

- Authentication

  The source includes information in each sent message that identifies the source as authentic, and performs the required functions to ensure message integrity. A typical authentication scheme requires that the source and destination parties share the same authentication key.

- Privacy

  Messages are encrypted to achieve privacy. The encryption is done in such a way that only the intended destination can perform the decryption. A typical privacy scheme requires that the source and destination parties share the same privacy key.

- Access control

  Both the source and destination play a part in access control. Each destination may have a distinct access policy for each potential source, which gives an administrator considerable flexibility in setting up an SNMP management system and assigning various levels of authorization to different users.

EMC Smarts SNMP version support

The IP Manager fully supports SNMPv1 and v2c. It also provides the following support for SNMPv3:

- Supports SNMPv3 authentication, data encryption, and access control
- Supports manual discovery but not autodiscovery or topology import of candidate systems having SNMPv3 agents
- Authenticates and decrypts SNMPv3 traps

Controlling EMC Smarts ICMP and SNMP polling

The parameters for controlling IP Manager ICMP and SNMP polling during discovery are contained in the discovery.conf file and described in Chapter 2, "Setting Configuration Parameters," in the *EMC Smarts IP Management Suite Configuration Guide*.

The parameters for controlling ICMP and SNMP polling for correlation analysis are accessed through the Polling and Thresholds Console and are described in Chapter 6, "Customizing Polling," in the *EMC Smarts IP Management Suite Configuration Guide*.

C

Wildcard Patterns

This chapter shows you how to use a wildcard pattern in a text field to match a number of elements instead of specifying each element by name. (EMC Smarts programs, such as the Global Console, enable you to specify wildcard patterns in text fields.) It consists of the following sections:

- Types of wildcard patterns C-2

Types of wildcard patterns

A wildcard pattern is a series of characters that are matched against incoming character strings. You can use these patterns when you define pattern matching criteria.

Matching is done strictly from left to right, one character or basic wildcard pattern at a time. Basic wildcard patterns are defined in Table C-1. Characters that are not part of match constructs match themselves. The pattern and the incoming string must match completely. For example, the pattern abcd does not match the input abcde or abc.

A compound wildcard pattern consists of one or more basic wildcard patterns separated by ampersand (&) or tilde (~) characters. A compound wildcard pattern is matched by attempting to match each of its component basic wildcard patterns against the entire input string. For compound wildcard patterns, see Table C-2.

If the first character of a compound wildcard pattern is an ampersand (&) or tilde (~) character, the compound is interpreted as if an asterisk (*) appeared at the beginning of the pattern. For example, the pattern ~*[0-9]* matches any string not containing any digits. A trailing instance of an ampersand character (&) can only match the empty string. A trailing instance of a tilde character (~) can be read as "except for the empty string."

Note: Spaces are interpreted as characters and are subject to matching even if they are adjacent to operators like "&".

Table C-1 Basic wildcard patterns

Character	Description
?	Matches any single character. For example, server?.example.com matches server3.example.com and serverB.example.com, but not server10.example.com.
*	Matches an arbitrary string of characters. The string can be empty. For example, server*.example.com matches server-ny.example.com and server.example.com (an empty match).
[set]	Matches any single character that appears within [set]; or, if the first character of [set] is (^), any single character that is *not* in the set. A hyphen (-) within [set] indicates a range, so that [a-d] is equivalent to [abcd]. The character before the hyphen (-) must precede the character after it or the range will be empty. The character (^) in any position except the first, or a hyphen (-) at the first or last position, has no special meaning. For example, server[789-].example.com matches server7.example.com through server9.example.com, but not server6.example.com. It also matches server-.example.com. For example, server[^12].example.com does not match server1.example.com or server2.example.com, but will match server8.example.com.

Table C-1 Basic wildcard patterns (continued)

Character	Description
<n1-n2>	Matches numbers in a given range. Both *n1* and *n2* must be strings of digits, which represent non-negative integer values. The matching characters are a non-empty string of digits whose value, as a non-negative integer, is greater than or equal to *n1* and less than or equal to *n2*. If either end of the range is omitted, no limitation is placed on the accepted number. For example, 98.49.<1-100>.10 matches a range of IP addresses from 98.49.1.10 through 98.49.100.10. Example of an omitted high end of the range: <50-> matches any string of digits with a value greater than or equal to 50. Example of an omitted low end of the range: <-150> matches any value between zero and 150. For a more subtle example: The pattern <1-10>* matches 1, 2, up through 10, with * matching no characters. Similarly, it matches strings like 9x, with * matching the trailing x. However, it does not match 11, because <1-10> always extracts the longest possible string of digits (11) and then matches only if the number it represents is in range.
|	Matches alternatives. For example,"ab|bc|cd" without spaces matches exactly the three following strings: "ab", "bc", and "cd". A | as the first or last character of a pattern accepts an empty string as a match. Example with spaces "ab | bc" matches the strings "ab" and " bc".
\	Removes the special status, if any, of the following character. Backslash (\) has no special meaning within a set ([set]) or range (<n1-n2>) construct.
\\	Removes the special status of the characters (,), [,], <, and >. For example, the pattern \\(A\\) matches the string (A).

Special characters for compound wildcard patterns are summarized in Table C-2.

Table C-2 Compound wildcard patterns

Character	Description
&	"And Also" for a compound wildcard pattern. If a component basic wildcard pattern is preceded by & (or is the first basic wildcard pattern in the compound wildcard pattern), it *must* successfully match. Example: *NY*&*Router* matches all strings which contain NY and also contain Router. Example: <1-100>&*[02468] matches even numbers between 1 and 100 inclusive. The <1-100> component only passes numbers in the correct range and the *[02468] component only passes numbers that end in an even digit. Example: *A*|*B*&*C* matches strings that contain either an A or a B, and also contain a C.
~	"Except" for a compound wildcard pattern (opposite function of &).If a component basic wildcard pattern is preceded by ~, it *must not* match. Example: 10.20.30.*~10.20.30.50 matches all devices on network 10.20.30 except 10.20.30.50. Example: *Router*~*Cisco*&*10.20.30.*~10.20.30.<10-20>* matches a Router, except a Cisco router, with an address on network 10.20.30, except not 10.20.30.10 through 10.20.30.20.

Index

Symbols

.import files 2-11, 2-12, 9-4

A

AccessAddressFormat 6-3
AccessMode
    ICMPONLY 7-6
    ICMPSNMP 7-6
    SNMPONLY 7-6
Add Agent 4-8, 4-9
Adding devices 11-2
ARP table 2-19
Attribute
    DisplayName 3-4
    IsManaged 11-4, 11-5
    Mode A-13
    Name 3-4
AUTODETECT 6-3, 7-5
Autodiscovery 2-2, 6-10
    ARP table 2-19
    Enabling 4-6
    Filtering 7-2
    Filters 4-3, 7-10
    IP address table 2-19
    Safeguards 4-3
    Seed system 4-3
    Stopping 4-12
    Topology MIBs 2-19

B

Bridge A-10
Bridge probe 2-15

C

Cable A-13
Certification, device 2-11
Cisco Discovery Protocol (CDP) 2-16, 2-19, 6-10
Codebook radius 5-4
Community string 4-6, 6-6, 8-7, 8-10
    Adding 11-2, 11-3
Connection
    Cable A-13
    Network connection A-13
    Trunk cable A-13
Console
    Domain Manager Administration Console 5-4
    Notification Properties dialog 8-9
Containment probe 2-14
Correlation interval 5-4
Correlation radius 5-4
Creating
    IP tag filter groups 11-4
custom-end-post.asl 9-4
custom-end-system.asl 9-4
custom-start-fulldscv.asl 9-4
custom-start-post.asl 9-4
custom-start-system.asl 9-4

D

defaultReadCommunity 11-3
defaultSNMPPort 6-3, 7-5
Devices
    Removing from topology 11-9
Discovery
    .import files 2-11, 2-12, 9-4
    AccessMode 6-3, 7-6
    Add Agent 4-8, 4-9
    Adding a community string 11-2, 11-3
    AddressFormat 6-3
    AuthPass 6-4
    AuthProtocol 6-4
    autodiscovery 6-10
    Bridge probe 2-15
    Certification 2-11
        CERTIFIED 2-11
        GENERIC 2-11
        TEMPLATE 2-11
    Community string 4-6, 6-4, 6-6, 7-6, 8-7, 8-10
    Containment probe 2-14
    Context 6-5
    Definition of 1-2
    Discover All command 10-6
    Discover Pending command 10-6
    Discovery error scenarios 8-3
    Discovery errors 8-5

Discovery hook scripts
custom-end-post.asl 9-4
custom-end-system.asl 9-4
custom-start-fulldscv.asl 9-4
custom-start-post.asl 9-4
custom-start-system.asl 9-4
Discovery results 8-2
discovery.conf 8-15
DiscoveryError notification 8-9
dmctl 10-7
Domain Name System (DNS) 3-2, 8-5
Duplicate notification 11-7
EngineID 6-4
Excluding systems 7-3
Filtering 7-2
Filters 7-2
Full discovery interval 5-4, 10-4
GENERIC 9-2
Hook script files 2-12, 9-4
Host Resources probe 2-17
HSRP probe 2-15
Import from seed file 4-8
Importing topology 4-11, 7-2
IP Network probe 2-15
IP tagging 1-10
IPSec and IKE tunnels 1-12, 2-20
IPV6 probe 2-15
LMHOSTS 8-6
Load balancers 1-13
Manual 2-2
Manual rediscovery 10-6
Methods 2-2
Name format
TM_USEAGENTADDRESS 6-4
TM_USEAUTONAME 6-4
TM_USELOOPBACK 6-4
TM_USENONPRIVATEIP 6-4
TM_USEPRIVATEIP 6-4
TM_USESEEDNAME 6-4
TM_USESYSNAME 6-4
Name Resolution probe 2-13, 3-4
NameFormat 6-4
Neighbor probe 2-16
NetBIOS 8-6
Node 8-3
oid2type files 2-11, A-19
OutOfLicense notification 2-8, 8-3, 8-8, 8-12
Pending Devices list 2-8, 4-4, 8-12, 10-2
Pending discovery interval 5-4, 8-13, 10-4
Performance probe 2-16
Phases of 2-5
Phase 1 2-5
Phase 2 2-6
Phase 3 2-8
Phase 4 2-18
PrivPass 6-5
PrivProtocol 6-5
Progress window 8-5
Rediscover command 10-6
Scheduled discovery interval 10-2
Seed file 6-2
Creating 6-12
Encrypting passwords in 6-13
Format 6-3
Importing 4-8
Sources 6-10
Seed file syntax 6-3
SeedName 6-4
Short discovery 2-20
SNMP timeout 8-9
SNMP version
AUTODETECT 6-3, 7-5
V1 6-3, 7-5
V2C 6-3, 7-5
V3 6-3
SNMPPort 6-3, 7-5
SNMPVersion 6-3, 7-5
System limit 4-4
System naming 3-4
TEMPLATE 9-3
THROTTLED 8-13
TM_USEAUTONAME 3-4
TM_USESEEDNAME 3-5
Topology import 2-2
UNDISCOVERED 8-13
Unsupported devices 8-7
UserName 6-4
Virtual Router probe 2-16
VLAN probe 2-14
WINS 8-6
Discovery error scenarios 8-3
Discovery errors 8-5
Discovery hook scripts
custom-end-post.asl 9-4
custom-end-system.asl 9-4
custom-start-fulldscv.asl 9-4
custom-start-post.asl 9-4
custom-start-system.asl 9-4
Discovery Progress window 4-4, 4-8, 4-12, 8-5, 8-8, 8-12
Discovery protocols
ICMP 1-3
SNMP MIB 1-3
Discovery results 8-2
discovery.conf 8-15
AccessAddressFormat 6-3
defaultReadCommunity 11-3
defaultSNMPPort 6-3, 7-5
ipExcludeList 7-3
MaximumHostBits 4-5
ShowPendingNONSNMP 8-3, 8-7, 8-13
DiscoveryError notification 8-9
DisplayName attribute 3-4
dmctl 10-7
dmctl utility 10-2
Domain Manager Administration Console 5-3
Discover All command 10-6
Discover Pending command 10-6
Discovery Progress window 4-4, 4-12, 8-5, 8-8, 8-12
Layout 5-4
Manage/Unmanage commands 11-6

Rediscover command 10-6
Save topology 11-10
Toolbar buttons 5-5
Domain Name System (DNS) 8-5
Duplicate notification 11-7

E

Encrypting passwords in seed file 6-13
Extreme Discovery Protocol (EDP) 2-16, 6-10

F

File server A-10
Filters
Examples of 7-7
Firewall A-10
Foundry Discovery Protocol (FDP) 2-16, 6-10
Full discovery 10-3
Full discovery interval 5-4, 10-4

G

GENERIC 9-2
Global Console
Overview discussion 5-2
Type of
Domain Manager Administration 5-2
Global Manager Administration 5-2

H

Hook script files 2-12, 9-4
Host A-10
Host Resources probe 2-17
HSRP probe 2-13, 2-15
Hub A-10

I

ICIM
Hierarchy A-4
Logical A-4
Physical A-4
ICIM_LogicalElement A-4
ICIM_PhysicalElement A-4
ICIM_PhysicalPackage A-4
ICMP
Definition of B-2
ICMPONLY 7-6
ICMPSNMP 7-6
Import from seed file 4-8
Interface A-12
Interface management policy 11-4
IP address table 2-19
IP interfaces
Unmanaging 11-5
IP Manager
Extracting seed file 11-8
Log file 8-5
Repository file 11-10
IP Network probe 2-15
IP product
Purpose 3-ix
IP tag filter groups 11-4
IP tagging 1-10
ipExcludeList 7-3
IPSec and IKE tunnels 1-12, 2-20
Discovering 2-20
Layering 2-20
IPV6 probe 2-15
IsManaged attribute 11-4, 11-5

L

Layering 2-20
LMHOSTS 8-6
Load balancer 1-13, A-10
Log files 1-5, 8-5
Loss symptom probability 5-4

M

Manage command 11-6
Managing topology objects 11-4
Manual discovery 2-2
Mark Non-SNMP 8-14
Matching
Pattern C-2
MaximumHostBits 4-5
Mode attribute A-13
Multilayer switch feature card (MSFC) A-10

N

Name attribute 3-4
Name resolution
Automatic 3-7
Configuring 3-6
Seed name 3-9
NameFormat 3-4
TM_USEAUTONAME 3-4
TM_USESEEDNAME 3-5
Naming a system 3-4
Neighbor probe 2-16
Neighboring system 2-16
NetBIOS 8-6
Network adapter
Interface A-12
Mode A-13
Port A-12
Sub-interface A-12
Network connection A-13
Node 8-3, A-10
Notification
DiscoveryError 8-9
Duplicate 11-7
OutOfLicense 2-8, 8-3, 8-8, 8-12
Notification Properties dialog 8-9
Number of problems 5-4

O

oid2type files 2-11, A-19
oid2type_Field.conf 9-2

Operator
Wildcard C-2
OutOfLicense notification 2-8, 8-3, 8-8, 8-12

P

Pattern C-2
Pattern matching C-2
Pending Devices list 2-8, 4-4, 8-12, 8-13, 10-2
Accept 8-14
Contents description 8-13
How systems are added 8-12
Mark Non-SNMP 8-14
Rediscover 8-14
Remove 8-14
THROTTLED 8-3, 8-13
UNDISCOVERED 8-3, 8-13
Pending discovery 10-3
Pending discovery interval 5-4, 8-13, 10-4
Performance probe 2-16
Port A-12
Probe A-10
Probes
Bridge probe 2-15
Containment probe 2-14
Host Resources probe 2-17
HSRP probe 2-15
IP Network probe 2-15
IPV6 probe 2-15
Name Resolution probe 2-13
Neighbor probe 2-16
Performance probe 2-16
Virtual Router probe 2-16
VLAN probe 2-14

R

Rediscover command 10-6
Rediscovery 10-2
dmctl 10-7
Full discovery interval 10-4
Pending discovery interval 10-4
Scheduled discovery interval 10-2
Starting manually 10-6
Repository
Location 11-10
Router A-10
Router switch feature card (RSFC) A-11
Router switch module (RSM) A-11

S

Save topology 11-10
Scheduled discovery interval 10-2
Seed file 6-2
Creating 6-12
Encrypting passwords in 6-13
Extracting from the IP Manager 11-8
Format 6-3
Importing 4-8
Sources
CiscoWorks2000 6-10
TACACS 6-10
Syntax 6-3
Seed system 4-2, 4-3
Setting
Interface management policy 11-4
Short discovery 2-20
Configuring 2-20
ShowPendingNONSNMP 8-3, 8-7, 8-13
sm_rebond 6-13
sm_tpmgr 11-8
SNMP
AccessMode 6-3, 7-6
AddressFormat 6-3
AuthPass 6-4
AuthProtocol 6-4
Basic operations B-4
Community string 4-6, 6-4, 6-6, 7-6, 8-7, 8-10
Adding 11-2, 11-3
Context 6-5
Default port 7-5
Definition of B-3
EngineID 6-4
Key components B-3
NameFormat 6-4
PrivPass 6-5
PrivProtocol 6-5
SeedName 6-4
SNMPPort 6-3, 7-5
SNMPv1 security B-4
SNMPv2c security B-4
SNMPv3 and autodiscovery 6-11, 7-2
SNMPv3 and topology import 6-11, 7-2
SNMPv3 security B-4
SNMPVersion 6-3, 7-5
Supported versions B-5
Timeout 8-9
Trap
coldStart 8-13
warmStart 8-13
Unsupported devices 8-7
UserName 6-4
Versions B-4
SNMPONLY 7-6
SNMPv3
See SNMP
Spurious symptom probability 5-4
Sub-interface A-12
Switch A-11
sysObjectID 8-3, 8-4
System
Bridge A-10
DisplayName attribute 3-4
File server A-10
Firewall A-10
Host A-10
Hub A-10
Load balancer A-10
Multilayer switch feature card A-10
Name attribute 3-4
Naming 3-4

Neighbor 2-16
Node A-10
Probe A-10
Router A-10
Router switch feature card A-11
Router switch module A-11
Switch A-11
Terminal server A-11
Uncertified A-11

T

TEMPLATE 9-3
Terminal server A-11
THROTTLED 8-3, 8-13
TM_USEAUTONAME 3-4
TM_USESEEDNAME 3-5
Topology
Autodiscovery 2-2
File location 11-10
Importing 4-11, 7-2
Importing from adapter 2-2
Managing and unmanaging 11-4
Manual save 11-10
Removing systems 11-9
Save Topology command 11-10
Saving to file 11-10
Trunk cable A-13

U

Uncertified A-11
UNDISCOVERED 8-3, 8-13
Unmanaging
IP interfaces 11-5
Topology objects 11-4

V

Virtual Router probe 2-16
VLAN probe 2-14

W

Wildcard 7-7, C-2
Chart of operators C-2
WINS 8-6